STOCHASTIC PREDICTION AND FEEDBACK CONTROL OF ROUTER QUEUE SIZE IN A VIRTUAL NETWORK ENVIRONMENT

THESIS

Muflih Alqahtani, First Lieutenant, RSAF

AFIT-ENG-T-14-S-10

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT-ENG-T-14-S-10

STOCHASTIC PREDICTION AND FEEDBACK CONTROL OF ROUTER QUEUE SIZE IN A VIRTUAL NETWORK ENVIRONMENT

THESIS

Presented to the Faculty

Department of Electrical and Computer Engineering

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the

Degree of Master of Science in Computer Engineering

Muflih Alqahtani, BS

First Lieutenant, RSAF

September 2014

AFIT-ENG-T-14-S-10

STOCHASTIC PREDICTION AND FEEDBACK CONTROL OF ROUTER QUEUE SIZE IN A VIRTUAL NETWORK ENVIRONMENT

Muflih Alqahtani, BS

First Lieutenant, RSAF

Approved:

______________//signed//______________	22 August 2014
LTC Robert J. McTasney, PhD (Chairman)	Date
______________//signed//______________	22 August 2014
Dr. Kenneth M. Hopkinson, PhD (Member)	Date
______________//signed//______________	26 August 2014
Dr. Robert F. Mills, PhD (Member)	Date

AFIT-ENG-T-14-S-10

Abstract

Modern congestion and routing management algorithms work well for networks with static topologies and moderate QoS requirements. However, these algorithms may not be suitable for modern military networks with fluid dynamic topologies and traffic demands that include mobile networks with many assets. These secure networks require a high level of Quality of Service (QoS) that must adapt to changing demands with no service interruptions. The idea of a network weatherman was developed by Stuckey to predict router queue size by using an extended Kalman filter [2]. He modeled and exercised his design in OPNET with positive results. The goal of this research is to investigate the use of queue size predictions and network weatherman and to determine the effectiveness of three types of filters to predict future traffic demand in a virtual network environment. These filters are an extended Kalman filter, an unscented Kalman filter, and a basic filter. These queue size predictions will be used to implement a network controller to improve the performance of information technology (IT) networks and formulate some type of context awareness and cognitive process in the management of networks by reducing packet loss.

To My parents

Brothers and Sisters

Acknowledgments

I would never have been able to finish my thesis without the help of ALLAH and then the guidance of my committee members, and support from my friends and family.

I would like to express my sincere appreciation to my faculty advisor, LTC McTasney, for his excellent guidance, caring, patience, and support throughout the course of this thesis effort. The insight and experience was certainly appreciated. I would like to thank Capt. Joah A. Betances with the assistance of understanding the previous work. I would like to show appreciation to Capt. Humood Alkhaldi for his assistance in the understanding and programming of Kalman filter. Additionally, I would also like thank everyone in the Cyber Advanced Networking in Mobile Applications Laboratory (ANiMaL) for their assistance and support provided to me in this endeavor.

Finally, I would like to thank all my friends especially Capt. Saleh Alnatifat and Capt. Mohammed Alghofaily. They were always supporting me and encouraging me with their best wishes.

Muflih M. Alqahtani

Table of Contents

Page

List of Figures

List of Tables

STOCHASTIC PREDICTION AND FEEDBACK CONTROL OF ROUTER QUEUE SIZE IN A VIRTUAL NETWORK ENVIRONMENT

1. Introduction

Today, modern military networks and other sensitive organizational communications take place in an open environment that include mobile networks which have many assets, numerous elements and dynamic network topologies. These networks need to be secured and require a high level of Quality of Service (QoS) to prevent such organizations from service interruptions. Also, they must be able to adapt to changing demands. In the past, this ability was not as important because these networks were smaller with relatively static topologies.

Although, modern congestion and routing management algorithms work well for networks with static topologies and moderate QoS prerequisites, they are insufficient for military systems which dynamically change networks that need to be flexible to accomplish mission goals and require strict QoS. This requirement for strict QoS is due to the critical mission support they provide, for the most part in a mobile setting. Even with the advent of visualization tools, computer networks tend to be obscure to system administrators and the users since modern networking technology constrains the ability of a network to adjust automatically to the changing demands, often leading to sub-optimal performance. The use of bleeding-edge technologies with limited scope is considered “not worth the risk” with regard to security versus the benefit of improved performance.

Another problem is that layered protocol design suppresses the communication of network state information, where a crossed-layered approach can have more of an impact on improving network performance. Through cross-layered mechanisms, network administrators or their automated tools can improve performance, improving customer experience. The cross-layer mechanisms would facilitate the identification and resolution of changing traffic demands in terms of data flow and adjusting routing parameters.

If a smart agent could predict changes in the traffic demands to a network based upon the network's current state and had the ability to adjust network parameters, both through cross-layer mechanisms, the network could reach the goal of dynamically adapting to changing demands. This research makes a valiant attempt to reach this goal.

1.1 Objectives

The goal of this research is to determine the effectiveness of three types of filters to predict future traffic demand in a **virtual network environment**. These types of filters are: a basic filter, an extended Kalman filter, and an unscented Kalman filter. The next step is to implement a network controller to use these traffic demand predictions, to improve the performance of information technology (IT) networks and formulate some type of context awareness and cognitive process in the management of networks.

1.2 Contributions

This research combines several prior research efforts. The past research implemented a network controller that predicts traffic demand through measuring router queue size using an extended Kalman filter only [2]. These network models were implemented using network simulation platforms such as OPNET and NS2. However, these discrete-event network simulations and their models, had limitations. This research extends the previous simulation based modelling work into a virtual network environment where the degree of traffic demand is more representative of real traffic demand with the inclusion of real operating systems and applications versus application models provided by discrete-event modeling and simulation platforms conducted in the past work [2]. Scenarios with more realistic traffic loads and two new filters (the basic and the unscented Kalman filter) allow for better evaluation of the extended Kalman filter's effect on predicting and improving network performance. This more realistic evaluation and building of a new controller using additional filter algorithms will be my research contribution.

1.3 Thesis Overview

- Chapter 1: A brief introduction to the subject area of congestion and routing management algorithms and modern networks that require high level of QoS and highlight some of the inherent challenges in this type of network environment. Also, this chapter outlines my thesis objectives and contributions.
- Chapter 2: Provides background information and literature review that show the previous research that has been conducted.
- Chapter 3: Describes the methodology and the approach in detail to conduct the experiments.
- Chapter 4: Provides the results and analysis of the experiments.
- Chapter 5: Provides conclusions and recommendations for future work.

2. Literature Review

The computer networking field demands the achievement of network-level goals despite augmenting network sophistication. Wireless networks have been following a trend of progressively dynamic and diverse environments. Nevertheless, Air Force Institute of Technology researchers have continued investigating a fundamental new model to accomplish, dynamically and independently, the objectives of a multifaceted network in a wireless system. Distributed intelligence networks use system cognition, which is the efficiency of perceiving contemporary conditions, and then preparing, deciding, and working on them to formulate decisions that consider continuous objectives [1]. The cognitive system agents collaborate in a peer-peer approach to generate an intellectual distributed network. Advancements in distributed routing, wide-area system monitoring, and intelligent network optimization would ensure the realization of this visualization.

Figure 2.1 illustrates the combination of user preferences and the current working conditions combined with network objectives in the determination of the best network that meets all the user requirements. This basic outline does not perform complex intelligent commands, but is rather efficient when used for specialized organizations such as the military in predicting and controlling information flow [2]. The above system-level description serves as the driving motivation for the previous work of Stuckey and Haught along with this work.

This chapter provides background information starting with network queues, network information flow in networks of queues, Kalman filters, the Extended Kalman filter, the Unscented Kalman filter, and applications of Kalman filters in networking. Then, it demonstrates the previous work done by Stuckey and Haught and using network predictions to improve network performance by applying the dynamic routing queue controller implemented by Haught. It ends by presenting the tool that will be used in this research which is VMware workstation.

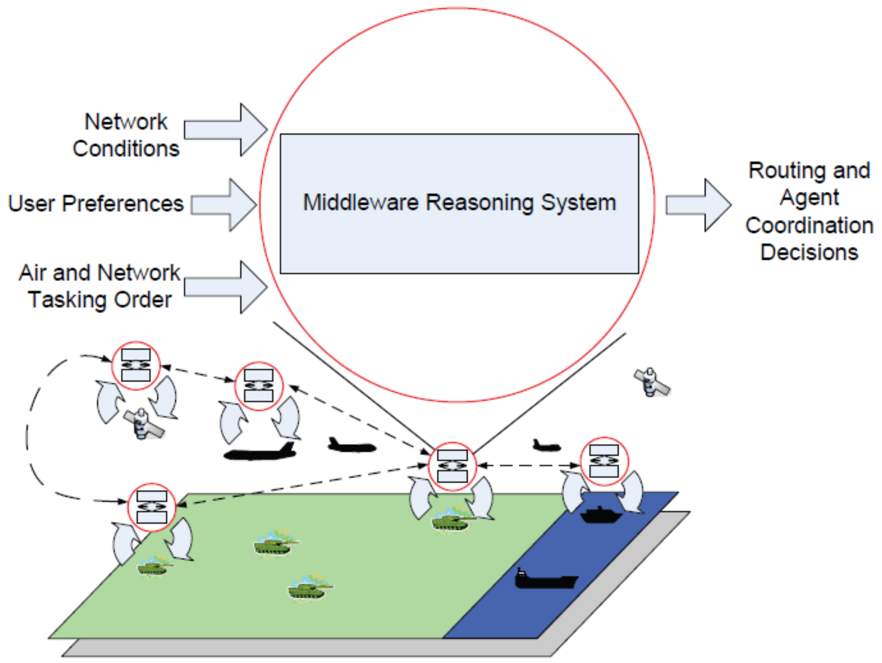

Network Conditions
User Preferences
Air and Network Tasking Order
Middleware Reasoning System
Routing and Agent Coordination Decisions

Figure 2.1: Determination of the best network that meets all the user requirements. [1]

2.1 Network Queues

The mathematical study of queues or waiting lines is referred to as queuing theory. It consist of models constructed with the aim of predicting waiting times and queue lengths. Queuing theory originated from Agner Erlang’s research in his attempt to describe the Copenhagen telephone exchange using created models [6]. Since then, the ideas have resulted in the emergence of applications such as computing, traffic engineering, and telecommunication.

In addition, queuing theory is considered a branch of operations research due to the use of its results in making business decisions concerning the resources required in service provision. However, the main and important application of queuing theory is in queuing networks, also known as networks of queues.

The main use of networks of queues is to model queuing and potential connections in cases of a shared set of resources. It is possible to model these types of networks using a set of service centers whereby each service center is not limited to one server [7]. A network of queues is composed of a collection of customers, which represents packets, and service centers, which represent servers. Consider a simple model in Figure 2.2, which has a single service center, where customers arrive at the service center, wait in the queue when necessary, receive service from the server when it is available, and then depart. This model has two parameters: specification of the service demand, which is the average service rate of a customer, and specification of the traffic demand, which is the arrival rate of customers.

Also, there are two types of queuing networks, open and closed queuing networks. Closed queuing networks are stationary since jobs neither enter nor depart from these networks. On the other hand, open queuing networks are further divided into two categories: open feedback queuing networks and open feed forward queuing networks. In an open feedback queuing network, a job may reenter the same queue that previously served it. However, in an open feed forward queuing network, a job can only appear in the same queue once.

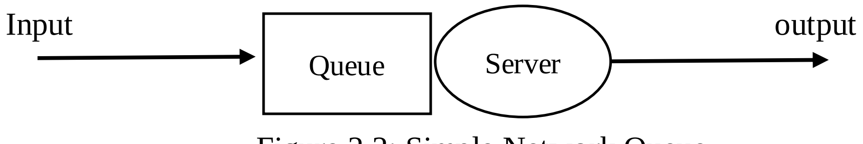


Figure 2.2: Simple Network Queue

The above discussion is pretty rudimentary, but it provides a foundation of understanding network information flow in networks of queues.

2.2 Network Information Flow in Network of Queues

The routing performance objectives with the network flow of information is an important field in networking research. Performance objectives range from quality of service such as delay and loss to high throughput depending on the application. Existing models which include a network of queues and network information flow both have their own advantages and disadvantages. However, a computer network can be better represented by combining the active features from these two models. In research on computer networks, the network flow model has been adopted for modelling unicast routing whereby the data streams flow from one particular sender to one receiver. On the other hand, there is the structure of multicast routing where data flows from one particular sender to multiple recipients. However, this structure is not categorized under the simple network flow category since flow conservation does not hold and information may be replicated.

Queuing theory provides a framework for modelling with limited capacities at network nodes using standard queuing models [8]. These queuing models provide methods for evaluating the average behavior of a node based on the amount of workload it is subjected to. Networks of queues can be applied to general network topologies in order to perform optimal multi-hop in routing of information flows. However, this is different from the traditional link cost based network flow models. This is because the new network of queues model contrasts most studies on network information flow since it considers nodes with finite processing capacities. These finite processing capacities are important to consider when determining optimal routes for network flows. In addition, they can have implications for the performance of network flows especially in terms of delay. The results from queuing theory are used to determine optimal routes for network flow that take into account the limited processing capacity of nodes. As a result, this computer network model captures two very important features. These features are finite node capacities and multi-hop routing of information flows as well as the associated queuing delays.

The queuing system is key point in a networking system. In real application, the queue size is limited and plays an important role in the performance, delay, and throughput of the network system. As a result, trying to predict the queue size in the future using a filter such as an extended or unscented Kalman filter will impact the performance of a network.

2.3 Kalman Filter

Kalman holds the status of being one of the important and commonly fused algorithms that are still in use today. The success of the Kalman filter has been from its small computational requirements, elegant recursive properties and its status of being the optimal estimator for linear and non-linear systems with Gaussian error statistics. Some of the typical uses of the Kalman filter includes the smoothening of noisy data and estimation of parameters of interest. Such applications are applied to global positioning system (GPS) receivers and inertial navigation systems (INS). Theoretically, the Kalman filter is an algorithm which permits correct deduction in a linearly and non-linearly changing system, which is a Bayesian model similarity to a hidden Markov model, but where the state of the dormant variables is indiscrete and where all dormant and observable variables have a Gaussian distribution [10].

The most famous early uses of Kalman filters were in the Apollo navigation computer that took astronauts to the moon and brought them back. Nowadays, Kalman filters are applied in every day satellite navigation devices, every Smartphone and others like computer games. The derivative of the Kalman filter is from the vector algebra as a minimum mean squared estimator. This is an approach which is suitable for students confident in mathematics but not one that is easy to grasp for students in mathematically heavy disciplines. Therefore, the Kalman filter is derived from the first principles considering the simple physical example exploiting a key property of the Gaussian distribution, specifically the property that the product of two Gaussian distributions is another Gaussian distribution [11].

2.3.1 Extended Kalman Filter

Generally, the problem of estimation of a state x of a discrete time controlled process is addressed by the Kalman filter. The above mentioned process is governed by the linear stochastic differential equation which is given by:

$$x_k = \mathbf{F}_k\, x_{k-1} + \mathbf{B}u_{k-1} + w_{k-1} \qquad (2.1)\ [2]$$

Where

- x : The estimate state.
- $\mathbf{F}$: The state transition matrix.
- $\mathbf{B}$: The control-input matrix
- u : The control input vector.
- w : The process noise vector.
- k : The time variable.

The measurement model z_k at time k of the state x_k is given by:

$$z_k = \mathbf{H}x_k + v_k \qquad (2.2)\ [2]$$

Where $\mathbf{H}$ is the measurement matrix and v_k is measurement noise vector. The process noise vector w_k and the measurement noise vector v_k are uncorrelated white Gaussian noises with zero mean and covariances given by

$$\mathbf{Q} = \mathbf{E}[w_k w_k^T] \qquad (2.3)\ [2]$$

$$\mathbf{R} = \mathbf{E}[v_k v_k^T] \qquad (2.4)\ [2]$$

Where **Q** is the process noise covariance (the error due to process) and **R** noise measurement covariance (the error due to measurements).

A common usage of Kalman filter, is to estimate a process through a system of feedback control. The filter provides an estimation of the process state at some time, consequently getting the response in the form of (noisy) measurements. There are two steps in Kalman filter; time update equations (prediction) and measurement correlation equations (update). The time update equations are in charge of forwarding (in time) the present state and error covariance estimation to obtain the priori estimates for the next time step. The measurement update equations are used in of the feedback. In this case they incorporate a new measurement into the priori estimate to get an improved posteriori estimate. The equations for update can as well be thought of as predictor equations, while the update equations for measurement can be thought of as corrector equations.

2.3.1.1 Predict Equations

In the prediction phase, there are two predicted equations; the priori state estimate, and the priori covariance estimate which are given by

$$\hat{x}_{\cdot k|k-1} = \mathbf{F}_k \hat{x}_{\cdot k|k-1} + \mathbf{B} u_k \qquad (2.5)\ [2]$$

$$\hat{P}_{\cdot k|k-1} = \mathbf{F}_k \hat{P}_{\cdot k|k-1}\ \mathbf{F}_k^T + \mathbf{Q}_k \qquad (2.6)\ [2]$$

Where $\hat{x}_{\cdot k|k-1}$ is a posteriori state estimate at time k and $\hat{P}_{\cdot k|k-1}$ is a posteriori error covariance matrix. $\mathbf{F}_k$ is the state transition model and $\mathbf{B}_k$ is applied to the control input vector u_k which is the control-input model. The process noise covariance is $\mathbf{Q}_k$.

2.3.1.2 Update Equation

Update innovation residual:

$$\tilde{y}_{\cdot k} = z_k - \mathbf{H}_k \hat{x}_{\cdot k|k-1} \qquad (2.7)\ [2]$$

Update innovation covariance:

$$S_k = \mathbf{H}_k P_{k|k-1} \mathbf{H}_k^T + \mathbf{R}_k \qquad (2.8)\ [2]$$

Kalman gain:

$$K_k = P_{k|k-1}\ \mathbf{H}_k^T S_k^{-1} \qquad (2.9)\ [2]$$

Update the posteriori state estimate

$$\hat{x}_{\cdot k|k} = \hat{x}_{\cdot k|k-1} + K_k \tilde{y}_{\cdot k} \qquad (2.10)\ [2]$$

Update the posteriori error covariance

$$P_{k|k} = (I - K_k \mathbf{H}_k) P_{k|k-1} \quad (2.11)\ [2]$$

2.3.2 Unscented Kalman Filter

The UKF is based on the principle of unscented transformation, which is the transformation of a set of points, or called sigma points, using a nonlinear function as shown in Figure 2.4 [19].

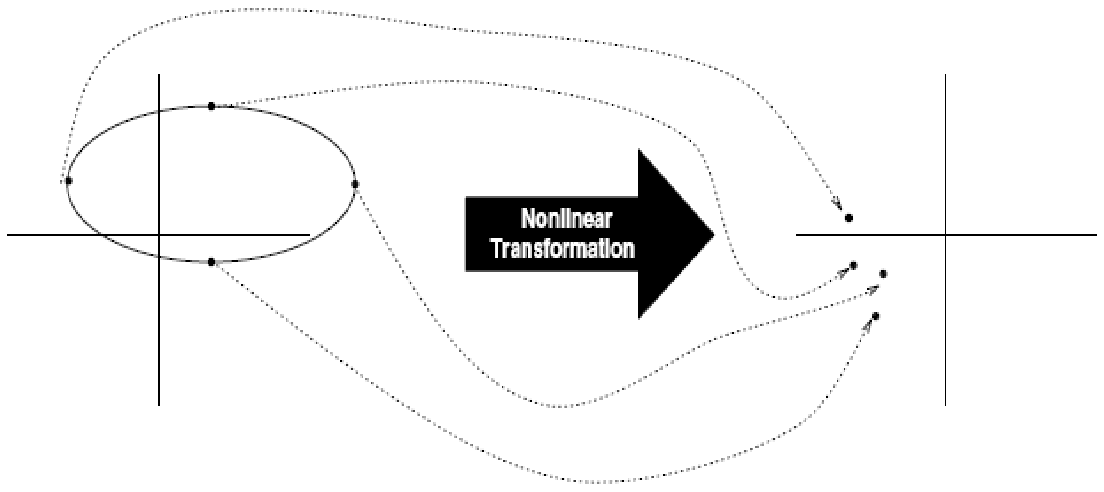


Figure 2.3: Basic Idea of Unscented Transform [19]

For instance, χ is a random variable with dimension n and has mean $\bar{x}$ and covarianceP_{xx}. In order to calculate the transformed sigma points $\mathcal{Y}_i$ through a nonlinear function

$y = f(x)$, we establish a matrix χ with dimension $2n + 1$ sigma vectors χ_i with corresponding weights W_i based on a well-specified algorithm which is given by the following equations:

$$\chi_0 = \bar{x} \qquad W_0 = \frac{k}{n+k} \tag{2.12}$$ [19]

$$\chi_i = \bar{x} + \left(\sqrt{(n+k)P_{xx}}\right)_i \qquad W_i = \frac{1}{2(n+k)} \tag{2.13}$$ [19]

$$\chi_{i+n} = \bar{x} - \left(\sqrt{(n+k)P_{xx}}\right)_i \qquad W_{i+n} = \frac{1}{2(n+k)} \tag{2.14}$$ [19]

Where

- χ : The sigma points vector.
- W : The associated weight of the sigma points.
- n : The dimension of the sigma vector.
- k : A scaling parameter.
- P_{xx} : The covariance of the sigma points.
- $\bar{x}$: The mean of the sigma points.

After calculating these sigma points with corresponding weights, they are passed and propagated through the nonlinear function $y = f(x)$ resulting into a transformed sigma points according to the following equations:

The transformed sigma points are

$$\mathcal{Y}_i = f\,[\chi_i] \tag{2.15}$$ [19]

The weighted mean of the transformed sigma points is

$$\bar{y} = \sum_{i=o}^{2n} W_i \, \mathcal{Y}_i \qquad (2.16)\ [19]$$

The weighted covariance of the transformed sigma points is

$$P_{yy} = \sum_{i=o}^{2n} W_i \, \{\mathcal{Y}_i - \bar{y}\}\{\mathcal{Y}_i - \bar{y}\}^T \qquad (2.17)\ [19]$$

The UKF performs exactly like or better than the extended Kalman filter but without the need to calculate Jacobeans of the nonlinear functions in the system. The ease of implementation and the high estimation accuracy of the unscented transformation (UT), makes it a better filtering/estimation algorithm than the Extended Kalman filter (EKF) for nonlinear systems [19].

2.4 Applications of Kalman filter in networking

Kalman filter have been used in a number of ways. In 1960, Rudolf Kalman developed the Kalman filtering to tackle spacecraft routing problem for the Apollo Space Program. Today, Kalman filters can be used in the tracking of objects such as balls, faces, heads and hands; fitting Bezier patches to point data; economics; navigation; in computer vision applications such as stabilizing depth measurements, feature tracking, cluster tracking, fusion of data from radar, laser scanner and stereo cameras for depth and velocity measurement [12]. As seen from Figure 2.4, Kalman filter observed a system states through a set of measurements to provide the optimal estimate of the system state.

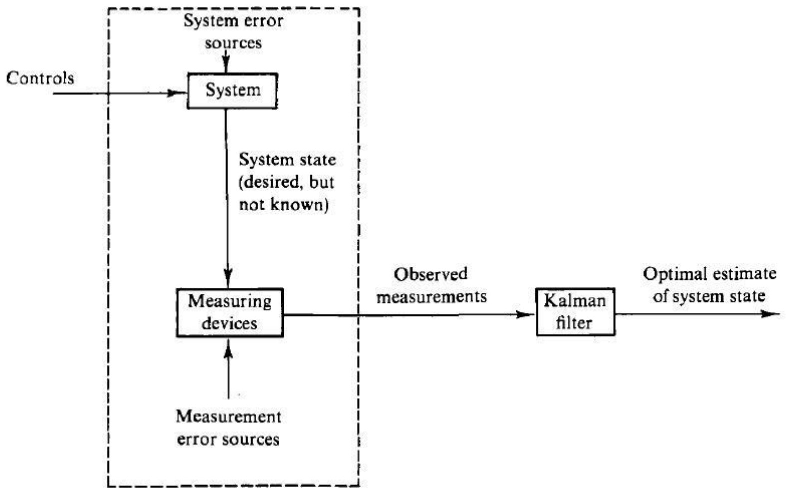

System error sources
Controls
System
System state (desired, but not known)
Measuring devices
Measurement error sources
Observed measurements
Kalman filter
Optimal estimate of system state

Figure 2.4: Typical Kalman filter application [12]

Also, there are many applications of Kalman filter in networking. In wireless Mobile networks, each mobile device is physically distributed and state updating of the mobile devices may generate a large amount of traffic saturating the network [12]. A Kalman filter is used to reduce the amount of traffic between devices to prevent bandwidth saturation. In addition, a Kalman filter has been used in wireless sensor networks (WSN). They maximize the capacity of a WSN by estimating the link quality [20]. They also have been used to estimate the optimal packet size to reduce the overhead cost [2].

This introduction of Kalman filters presented general and networking applications. The next section will present the use of the extended Kalman filter to predict the future state of a network.

2.5 Network Prediction

Nathan C. Stuckey developed the idea of a Network Weatherman (NWM) [3]. The NWM uses an algorithm to predict future state of the network based on the network's current state. Stuckey used an extended Kalman filter (EKF) to predict router queue size with respect to the existing and prior measurements of the queue size. These estimated predictions facilitated the management of network traffic improving throughput or reducing delays.

Stuckey integrated his NWM model into three OPNET network scenarios. The first scenario had a single one router network topology with one source node, one sink node and one router between them. The main aim of his first scenario was to apply the NWM practically and to validate the MATLAB modeled extended Kalman filter in an OPNET simulation. The second scenario had six source nodes to generate traffic, one sink node, and two routers as shown in the Figure 2.4. An extended Kalman filter is placed between the two routers to predict the second router (router 2) queue size. The presence of a projected queue size is important to optimize the network traffic.

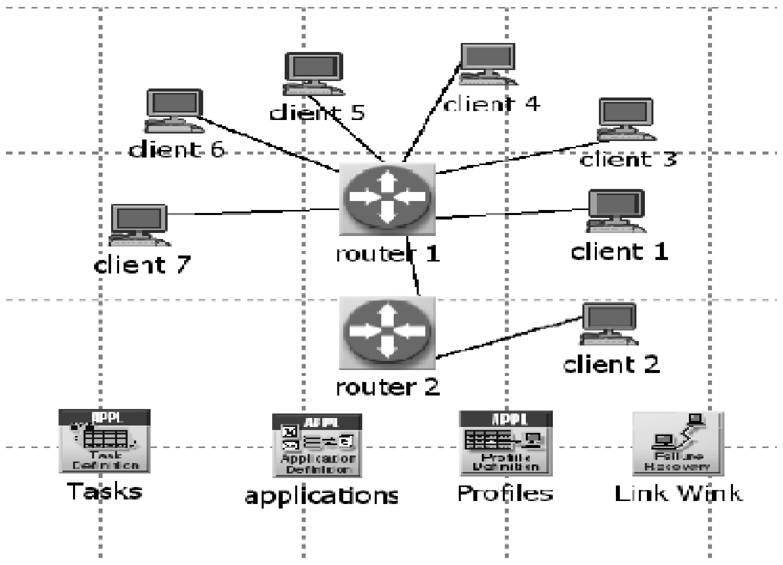


Figure 2.5: A Kalman filter placed between router 1 and router 2 in an OPNET simulation with numerous representative traffic types [3].

For third scenario, he implemented a simple network controller using the same network topology as the first scenario with the router queue size reducing the source's packet inter-arrival rate through feedback control. Although, he was able to estimate the queue size of the second router 10 seconds in the future as shown in Figure 2.5, the performance of his estimator was badly degraded when the predictions were too far in the future because there were small number of sampling of actual queue size provided to the Extended Kalman filter.

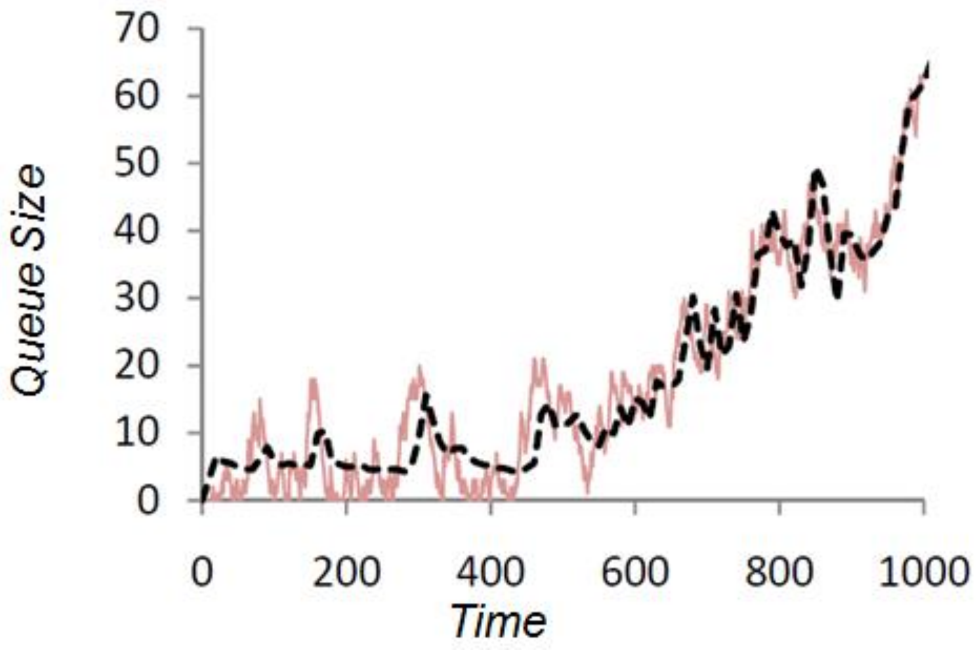


Figure 2.6: Actual queue size versus 10 second future queue size predictions of a Kalman filter placed between the two nodes in Figure 3 [3].

Stuckey's work was extended by James Haught, who tried to validate the efficiency of the extended Kalman filter prediction of the router queue size using NS2 network simulation. He performed several simulation scenarios using NS2 which consisted of many nodes with five extended Kalman filters as shown in Figure 2.6. In his experiments, every node was assigned various TCP and UDP flows with exponential arrival rates. The extended Kalman filters were able to accurately predict the network flow several seconds into the future, making a prediction every second for the five hundred seconds of simulation time. However, it was noted that predictions too far into the future were inaccurate [2].

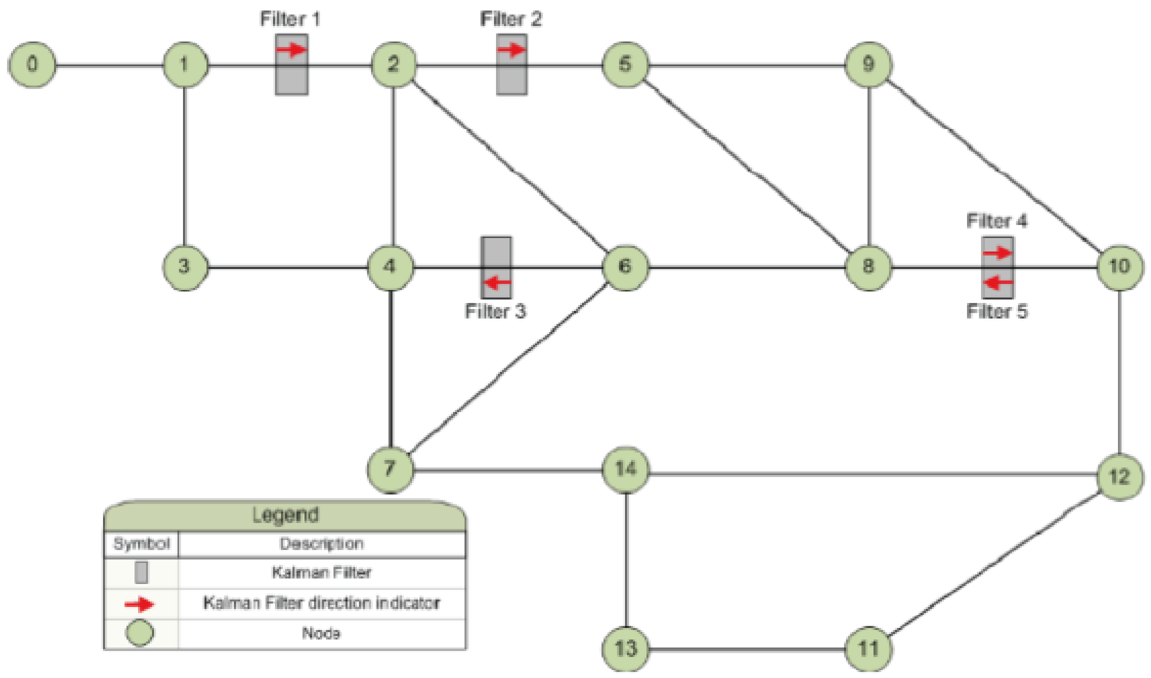


Figure 2.7: Kalman filter validation simulation network topology conducted by Haught [2]

After that, Haught developed Dynamic Routing Queue Controller (DRQC) to manage the network traffic. The DRQC adjusted network traffic based on current queue sizes as well as predicted queue sizes and network flows. The next section will describe in details the DRQC.

2.6 Dynamic Routing Queue Controller (DRQC)

The DRQC was a central network controller for routing traffic in networks based on the priority of the flow, network congestion, and queue prediction [2]. In the event of network congestion, the DRQC rerouted traffic to prevent the congestion based on the current and predicted router queue size. The DRQC used data from the NWM's projections to handle network congestion [3]. Large and medium networks with the DRQC possessed a decentralized architecture for regional and local controllers to tackle projections appropriately. The DRQC accessed queue sizes, prospective queue size predictions, and various network flows running in a network system. The key benefit of this centralized system was its ability to maintain accurately revised data lists that were easily accessible from every point. Unfortunately, a centralized solution is a solution with a single point of failure. In addition, Haught's networks were also affected by timing as opposed to small networks.

The DRQC obtained network states and made decisions for network optimization. This was to predict the level of congestion and to directly allocate network resources. Additionally, at the networks main routers, it repeatedly predicted and accessed the actual queue size at a constant rate, every 5 seconds [2]. The DRQC then made network control decisions based on these predicted and actual sizes of the router queue. If congestion was not detected, the DRQC made no changes except for rerouted or suspended network flow because of prior network congestion. Improving the network by regulating network flow routing or by reviving formerly suspended network flows was done by adjusting the network routing tables.

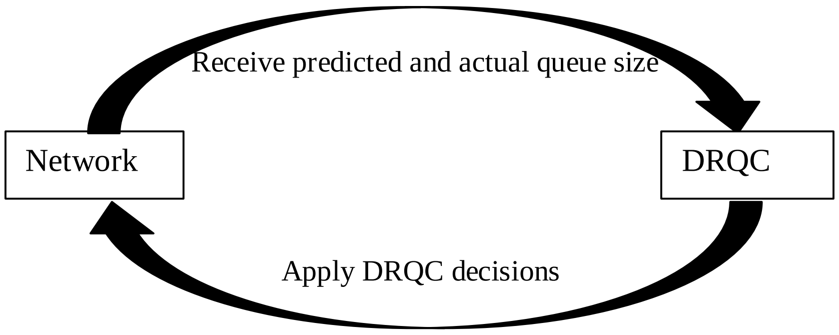


Figure 2.8: DRQC cycle [2].

Although, the main feature of the DRQC was to manage network traffic, it had other features. These major features of the DRQC included

a) Flow Reactivation
b) Priority Flow Control
c) Dynamically Split Flows
d) Priority Rerouting
e) Prediction Detection

Flow Reactivation: Dealt with the initial network flows that had earlier been delayed to avoid network congestion. If an existing dynamic flow completed data sending and suspension of network flow took place, the DRQC reassessed the network state to ensure the reactivation of other poised flows.

Priority Flow Control: this handled the first and the last flows in accordance to precedence. Lack of an available path for network flow with respect to the available connection abilities, flow suspension occurred to facilitate the passing of higher precedence flows [2].

Dynamically Split Flows: this facilitated the utilization of several paths for one flow to meet its bandwidth prerequisites. Most network routers failed to support split flows because it increases routing table’s complexity despite the fact that allowing split flows support fitting of additional flows.

Priority Rerouting: this transformed the paths in a network flow based on their precedence by altering the routing tables in that network. Use of flow ID supported different routing methods such as the MPLS and RSVP. Standard IP routing tables just consisted of the next-hop with respect to the objective in the header of the flow packets. DRQC’s routing tables facilitated the routing of packets with respect to flow ID. The flow ID paths in the networks were established, and controlled by the DRQC based on the flow preferences and existing bandwidth.

Prediction Detection: this was the ability of the controller to estimate router queue size through an Extended Kalman filter [2]. The assumption was that the controller had nonstop access to the real and predicted queue size information for the entire network.

In this research, Priority Rerouting and Prediction Detection has been applied to provide a better network performance.

2.7 VMware Workstation

VMware is a virtual machine that functions as a computer within a computer, which allows the hosting of an operating system within another operating system [13]. This is a system, which is installed in a host computer, where the host operating system VMware Workstation installation of other guest virtual machines. It also supports virtualization, in which the workstation is able to support other multiple programs installed on the host machine and at the same time allowing other multiple guest operating systems to run on the host. This model provides the best platform for virtualization.

Despite the many capabilities associated with VMware workstations, there are also disadvantages. VMware Workstation licenses are expensive. Another key important aspect to be taken into consideration are resource requirements. VMware workstations need additional resources, such as memory in order to support the applications running simultaneously. This raises the cost of setting up a VMware workstation [14], in which computers with limited speed and memory may need to be upgraded in order to enhance performance.

While there are disadvantages of using VMware, there are benefits gained from using the VMware workstation. The workstation is quite reliable and has greater performance. This is due to the hypervisor processor that the workstation is built using, which is able to handle memory complications while at the same time reduce bottlenecks that may be brought about by heavy workloads.

It also supports better administrative management. This is brought about by the console capability, and other features that allow easy management, such as patching and general administration of the machine. Another feature is the networking of many virtual machines on a single host. This can be supported by the virtual network features, which consist of bridged, NAT, and host-only that can be accessed through the virtual network editor. This capability allows building an IP addressing scheme through a DCHP server, which allows an IP address to be assigned to the workstation on the physical adapter through the DHCP server, while on the other hand NAT enables an IP address to be translated into a unique address in case there is a need to connect to the Internet. This allows building a simple network, with external capabilities that can stretch beyond the local environment.

VMware workstation has different network adapters, which include the host-only, bridged, and NAT. The host only allows the VMware workstation to connect only to another machine that is running the VMware workstation, while the bridged allows for connection through the adapter and the NAT uses a single external IP address that is not visible on the public network [15].

VMware workstation supports the drive to virtualization. There are technological infrastructures, such as datacenters that can be conceived from such innovation. It is a concept worth mentioning that has the potential to support future computing architectures. The concept has immense capabilities and can be a driving force towards cloud based technologies.

In this research, VMware workstation will be the tool used for building network topologies. These network topologies include virtual nodes as clients, servers, and routers. Host-only network adapters will be used to connect these virtual nodes while bridged adapter will allow access the virtual routers from host machine.

3. Methodology

Stuckey implemented a stochastic control scheme to estimate network state using an extended Kalman filter. He implemented the extended Kalman filter in MATLAB and made simulation models in an OPNET network simulation platform. On other hand, Haught developed a network estimator system and implemented a network queue controller using the same extended Kalman filter. He designed his estimator system using a different simulator; NS2, to validate the performance of Stuckey's estimator. Then he used this prediction system to implement a queue network controller to improve network performance (see Chapter II for more details). In this chapter, a new tool and models will be presented to validate Stuckey's network estimator and Haught's queue network controller.

3.1 Goals

The ***first goal*** of this research is to develop and validate Stuckey's stochastic control scheme in near-realistic network built in a virtual network environment (VMware workstation). Stuckey used OPNET simulation and an extended Kalman filter to validate his network estimator. The research methodology described in this section extends Stuckey's previous work by testing his methods using networked virtual machines using three different filters to estimate the network future states; a basic filter, an extended Kalman filter, and an Unscented Kalman filter. The ***second goal*** is to use these predictions of network state and apply them to the queue network controller to improve network performance. Different scenarios consisting of permutations of the prediction only and queue network controller feedback using each of the three filters here will be presented.

3.2 System Development

All of the experiments to test this technology will be done using VMware workstation. VMware workstation supports the creation of many guest virtual machines on a single host machine and offer the option of using combinations of different operating systems and applications. In the1960s, the concept of virtualization by IBM supported concurrent access to a mainframe computer [16]. For these experiments, Linux virtual clients, servers, and routers were created. The client generated traffic and communicated with a server through a router. MikroTik software was used to build the router to route the traffic between the client and the server. The routing functions are provided by the MikroTik RouterOS software which is an operating system based on the Linux Kernel to provide extensive stability, control, and flexibility for interfacing and routing [17].

Three different filters; basic filter, extended Kalman filter, and unscented Kalman filter, were written in MATLAB and executed in the host machine to predict network state. An FTP session between the MATLAB process on the host machine and the MikroTik RouterOS operating on VMware virtual machine, acting as a router, was used to pass the current queue state information. The filters used this current queue state data to predict the future of network state.

As seen in Figure 3.1, clients, RouterOS, and servers were created inside the host machine as virtual machines running Linux. Again, traffic from the client source would be routed through the virtual machine running RouterOS to the destination server. A script inside the RouterOS was executed every ‘x’ seconds to get the current queue state data and create a queue state’s file. In the host machine, the MATLAB script would establish the FTP connection to the router VM to get the queue state’s file which would include the current router queue size and the current time. At every prediction time, the MATLAB script would pass the current queue size and current time to the Kalman filter function to make the prediction of the future of queue network state; basically, predicting the queue’s future queue size.

Descriptions of the basic parts of the system starting from generating the traffic and ending with filters prediction of RouterOS queue size will be presented in the following subsections.

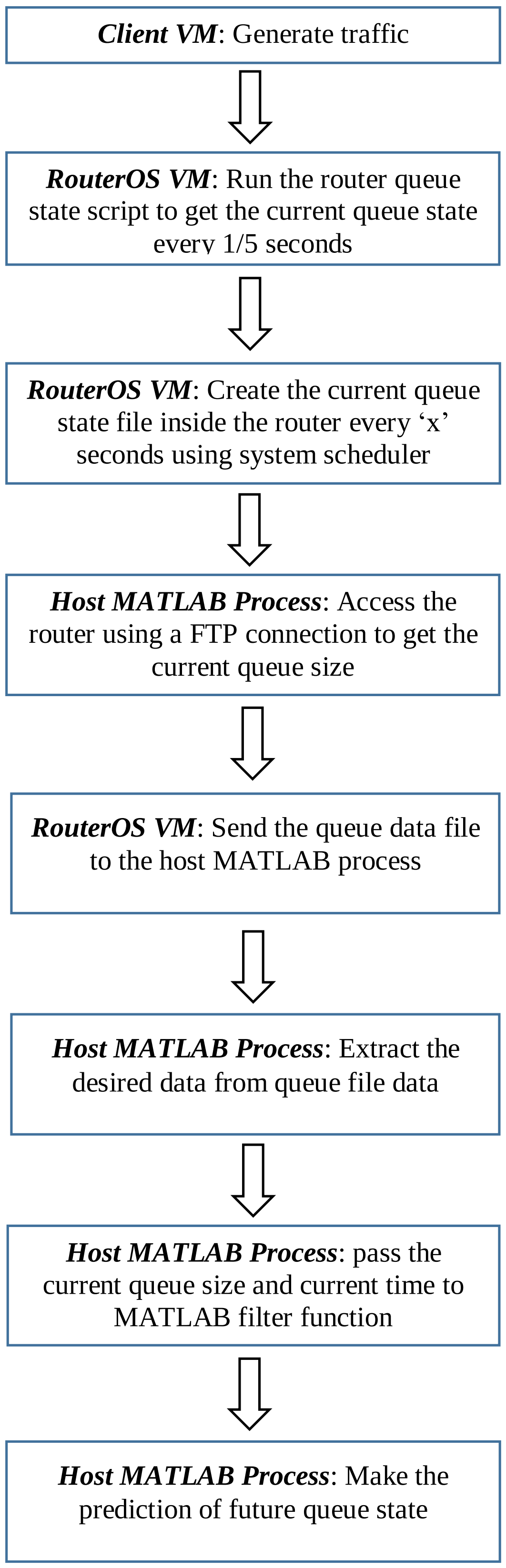

Client VM: Generate traffic
RouterOS VM: Run the router queue state script to get the current queue state every 1/5 seconds
RouterOS VM: Create the current queue state file inside the router every 'x' seconds using system scheduler
Host MATLAB Process: Access the router using a FTP connection to get the current queue size
RouterOS VM: Send the queue data file to the host MATLAB process
Host MATLAB Process: Extract the desired data from queue file data
Host MATLAB Process: pass the current queue size and current time to MATLAB filter function
Host MATLAB Process: Make the prediction of future queue state

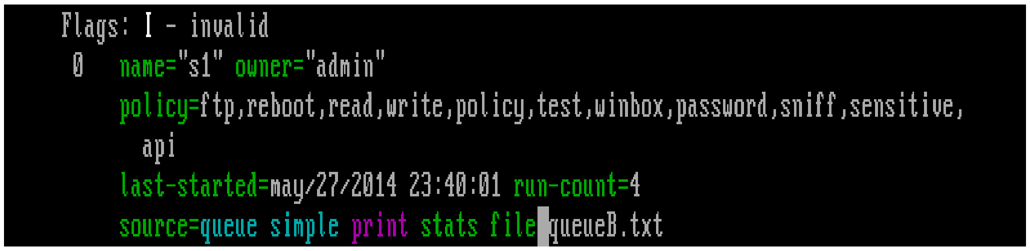
Figure 3.1: Overview chart of Simulation Approach

3.2.1 Generate the Traffic

UDP network sockets are used to generate the traffic load for experiments. In all simulations, connectionless socket datagrams are used to saturate the RouterOS queue.

3.2.2 Set up the Queue Router State Script

During each experiment, a script inside the RouterOS will be executed every 'x' seconds using RouterOS System Scheduler. This script will get the current state of the router queue and store it in a router queue state file inside the router. The host machine will then access the file through an FTP session. Figure 3.2 shows the queue state script.

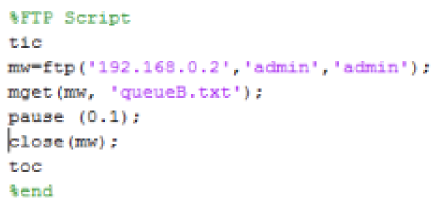


```
Flags: I - invalid
 0   name="s1" owner="admin"
     policy=ftp,reboot,read,write,policy,test,winbox,password,sniff,sensitive,
       api
     last-started=may/27/2014 23:40:01 run-count=4
     source=queue simple print stats file queueB.txt
```

Figure 3.2: RouterOS Queue State Script

3.2.3 Set up FTP Script

FTP script will be executed in the host machine every prediction time step to access the RouterOS and get the current queue state file and then pass it to the filter to predict the future state of the network.

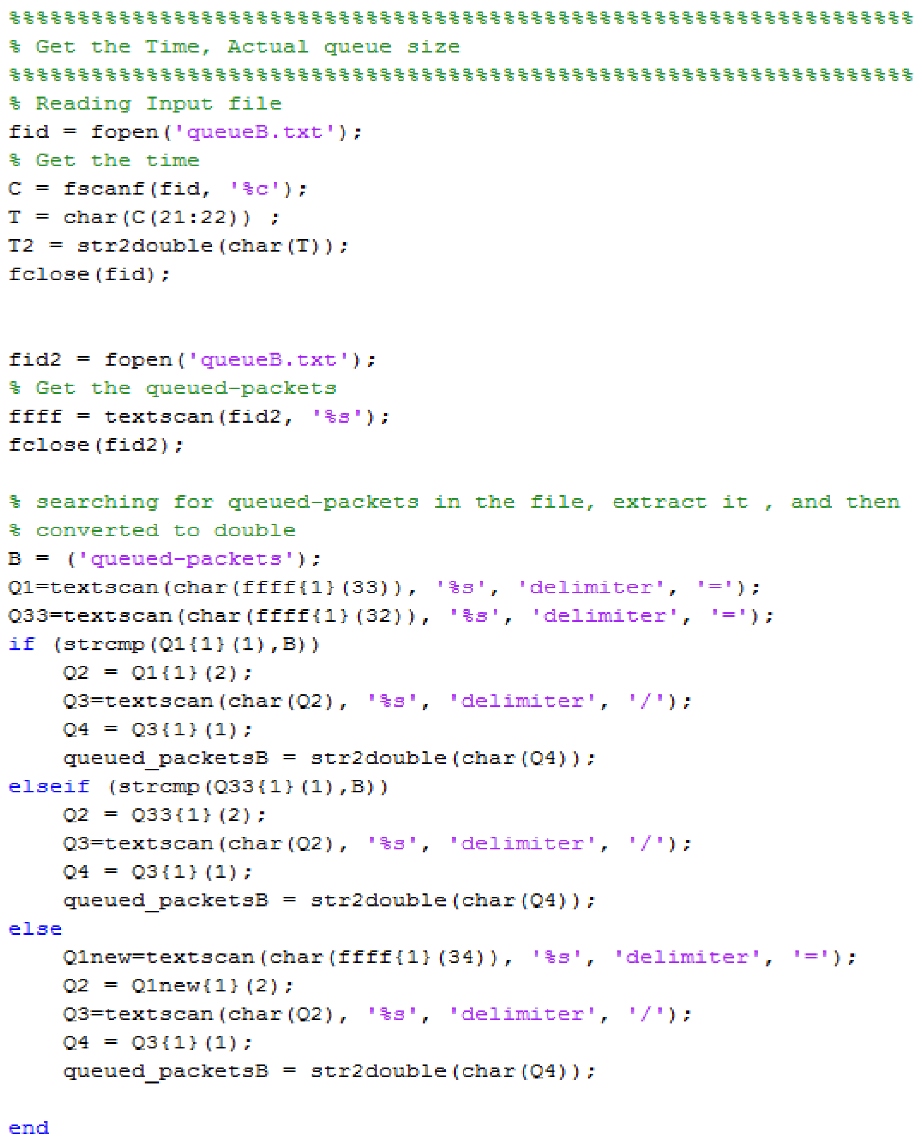


```
%FTP Script
tic
mw=ftp('192.168.0.2','admin','admin');
mget(mw, 'queueB.txt');
pause (0.1);
close(mw);
toc
%end
```

Figure 3.3: FTP Script

3.2.4 Set up the Queue Router State File

After the host machine receives the queue router state file, it will parse it and extract the queue size information which includes the current time the sample was taken and the size of the queue at that time. The script except that performs this action can found in Figure 3.4. This data will then be passed to the filter to make the prediction of the future state of the network queue.



```matlab
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Get the Time, Actual queue size
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Reading Input file
fid = fopen('queueB.txt');
% Get the time
C = fscanf(fid, '%c');
T = char(C(21:22)) ;
T2 = str2double(char(T));
fclose(fid);


fid2 = fopen('queueB.txt');
% Get the queued-packets
ffff = textscan(fid2, '%s');
fclose(fid2);

% searching for queued-packets in the file, extract it , and then
% converted to double
B = ('queued-packets');
Q1=textscan(char(ffff{1}(33)), '%s', 'delimiter', '=');
Q33=textscan(char(ffff{1}(32)), '%s', 'delimiter', '=');
if (strcmp(Q1{1}(1),B))
    Q2 = Q1{1}(2);
    Q3=textscan(char(Q2), '%s', 'delimiter', '/');
    Q4 = Q3{1}(1);
    queued_packetsB = str2double(char(Q4));
elseif (strcmp(Q33{1}(1),B))
    Q2 = Q33{1}(2);
    Q3=textscan(char(Q2), '%s', 'delimiter', '/');
    Q4 = Q3{1}(1);
    queued_packetsB = str2double(char(Q4));
else
    Q1new=textscan(char(ffff{1}(34)), '%s', 'delimiter', '=');
    Q2 = Q1new{1}(2);
    Q3=textscan(char(Q2), '%s', 'delimiter', '/');
    Q4 = Q3{1}(1);
    queued_packetsB = str2double(char(Q4));

end
```

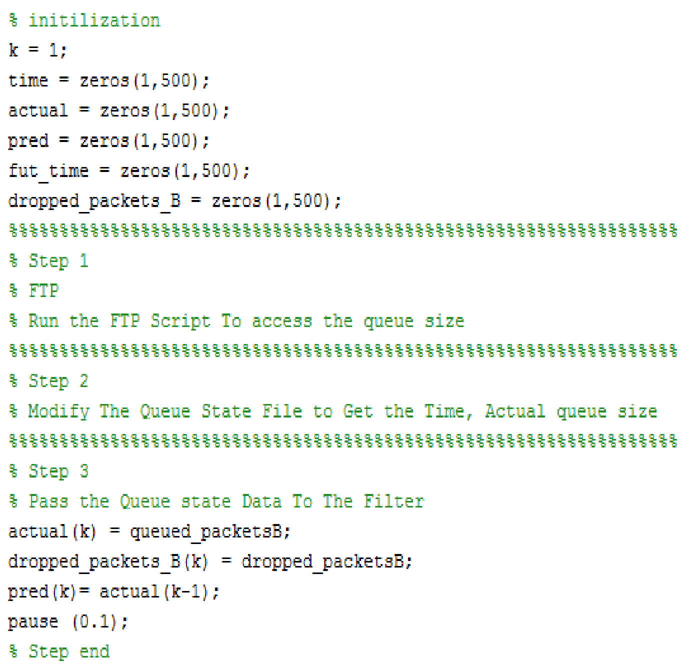
Figure 3.4: Queue state file manipulation

3.2.5 Filter design descriptions

Three filter design will be evaluated for each scenario. They are the Basic filter, the Extended Kalman filter and the Unscented Kalman filter. Each of these filters will be explained below in the following three subsections.

3.2.5.1 The Basic Filter

This section describes the Basic filter. The Basic filter is a baseline prediction filter that was used as a reference to evaluate the extended and the Unscented Kalman filters prediction performance. The Basic filter predicts the queue size based on the actual queue size of the previous queue sample received. The following figure is detailed description of the Basic filter. As seen in Figure 3.5, the current time and current actual queue size were received from RouterOS VM through the FTP session and will be stored in two MATLAB vectors; *time* and *actual*. At every prediction time, these vectors were passed to the Basic filter to predict the future queue state and calculated the dropped packets. The Basic filter stored the predicted queue size and dropped packets in *pred* and *dropped_packets* vectors, respectively.

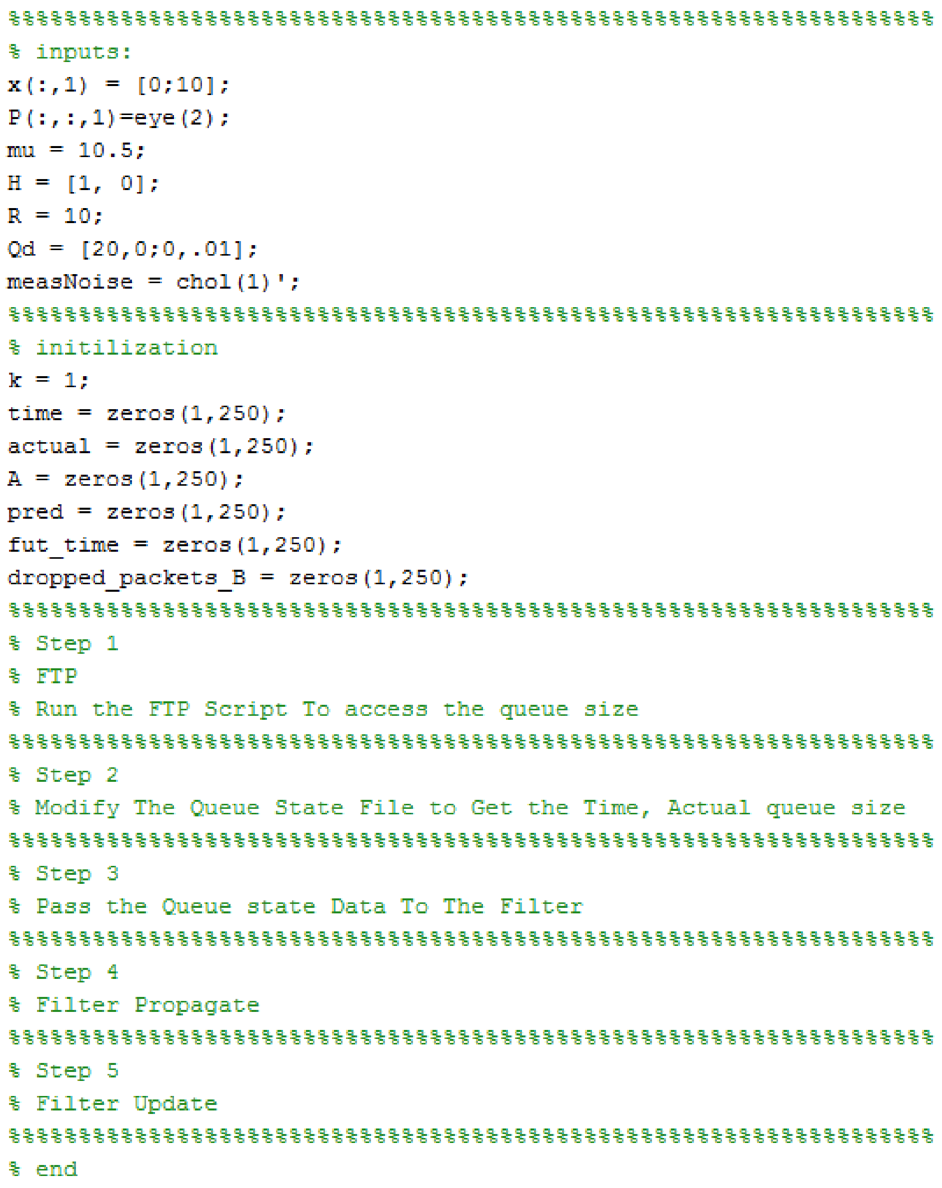


```
% initilization
k = 1;
time = zeros(1,500);
actual = zeros(1,500);
pred = zeros(1,500);
fut_time = zeros(1,500);
dropped_packets_B = zeros(1,500);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Step 1
% FTP
% Run the FTP Script To access the queue size
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Step 2
% Modify The Queue State File to Get the Time, Actual queue size
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Step 3
% Pass the Queue state Data To The Filter
actual(k) = queued_packetsB;
dropped_packets_B(k) = dropped_packetsB;
pred(k)= actual(k-1);
pause (0.1);
% Step end
```

Figure 3.5: Basic Filter MATLAB Code

3.2.5.2 The Extended Kalman Filter [3]

Stuckey developed an extended Kalman filter in MATLAB and integrated it into OPNET using a co-simulation interface to design his router's queue estimator [3]. To integrate this filter into the virtual network, only minor modifications for the purpose of interfacing, no functional changes, were made. As you can see from Figure 3.6, the input parameters were the same as inputs of Stuckey's extended Kalman filter. Three steps were added step1, step2, and step3. Step4 and step5, were the actual operations and equations of extended Kalman filter which were done in previous research (see Chapter II for more details). Step1 accessed RouterOS VM through an FTP session to receive the queue state file. After that, step 2 parsed this file and extracted the queue size information which includes the current sampling time which would be stored in in two vectors; *time* and *actual* . In step 3, this information, would be passed to the extended Kalman filter to predict future queue size. The predicted queue size and number of packets dropped results from the extended Kalman filter would be stored in *pred* and *dropped_packets* vectors respectively. The following Figure 3.6 is detail description of the modified extended Kalman filter.



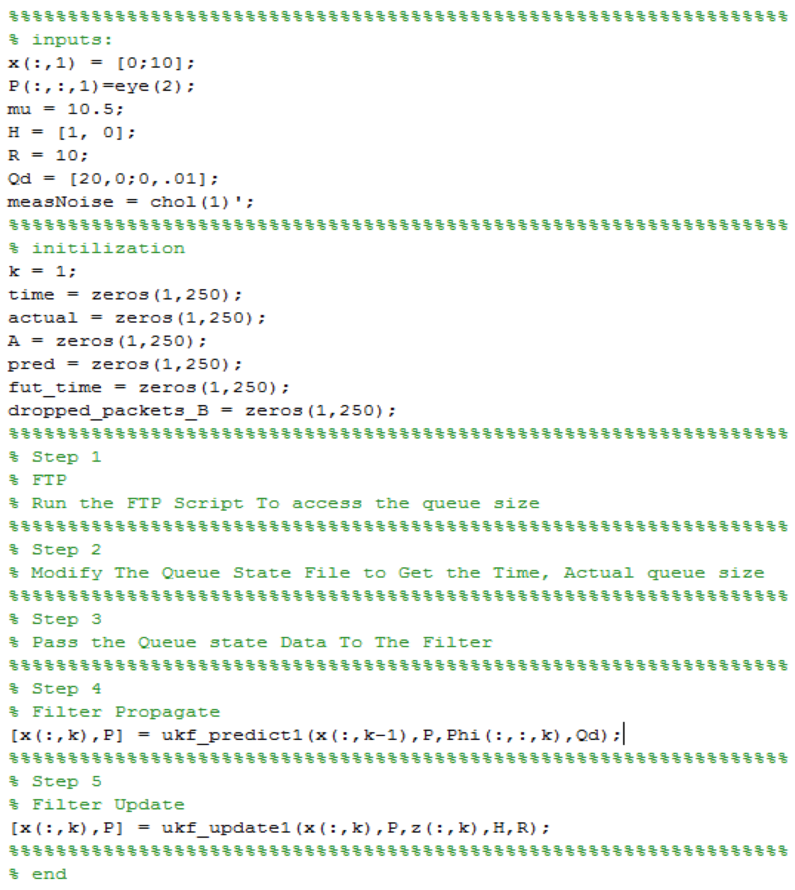
```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% inputs:
x(:,1) = [0;10];
P(:,:,1)=eye(2);
mu = 10.5;
H = [1, 0];
R = 10;
Qd = [20,0;0,.01];
measNoise = chol(1)';
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% initilization
k = 1;
time = zeros(1,250);
actual = zeros(1,250);
A = zeros(1,250);
pred = zeros(1,250);
fut_time = zeros(1,250);
dropped_packets_B = zeros(1,250);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Step 1
% FTP
% Run the FTP Script To access the queue size
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Step 2
% Modify The Queue State File to Get the Time, Actual queue size
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Step 3
% Pass the Queue state Data To The Filter
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Step 4
% Filter Propagate
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Step 5
% Filter Update
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% end
```

Figure 3.6: Extended Kalman Filter MATLAB Description

3.2.5.3 The Unscented Kalman Filter

The third filter used for network queue state predictor model was the Unscented Kalman filter (UKF). UKF is considered as the recent improvement of the Extended Kalman filter (EKF) [30]. In the Extended Kalman filter (EKF), the state distribution of the system is propagated analytically through the first-order linearization of the nonlinear system which may lead to errors to the posterior mean and covariance [30]. The UKF overcomes this problem by using a deterministic sampling approach which is representing the state of the system by minimal set of points called sigma points [30]. The UKF uses the 3rd order Taylor series expansion accuracy to capture the mean and covariance while the EKF achieves first order only [unscented]. (More details regarding equations of the EKF and the UKF in Chapter II)

The code of the Unscented Kalman was written by Simo in 2006 and updated and modified in 2010 by Jouni Hartikanen [30]. They implemented Bearing only Tracking (BoT) using the unscented Kalman filter. So, we modified Simo and Jouni code to build a network estimator using same unscented kalman filter. As you can see from figure 3.7, step 1 through step 3 were the same as extended Kalman filter in previous section. Step 4 and step 5 were the unscented Kalman filter functions that include many other MATLAB scripts.



```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% inputs:
x(:,1) = [0;10];
P(:,:,1)=eye(2);
mu = 10.5;
H = [1, 0];
R = 10;
Qd = [20,0;0,.01];
measNoise = chol(1)';
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% initilization
k = 1;
time = zeros(1,250);
actual = zeros(1,250);
A = zeros(1,250);
pred = zeros(1,250);
fut_time = zeros(1,250);
dropped_packets_B = zeros(1,250);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Step 1
% FTP
% Run the FTP Script To access the queue size
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Step 2
% Modify The Queue State File to Get the Time, Actual queue size
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Step 3
% Pass the Queue state Data To The Filter
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Step 4
% Filter Propagate
[x(:,k),P] = ukf_predict1(x(:,k-1),P,Phi(:,:,k),Qd);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Step 5
% Filter Update
[x(:,k),P] = ukf_update1(x(:,k),P,z(:,k),H,R);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% end
```

Figure 3.7: Unscented Kalman Filter MATLAB Description

3.3 Validation Post Processing vs Real Time Processing

To validate the model in a virtual network environment (VMware Workstation), a simple virtual client/server model that consist of client, RouterOS VM and server was made. The model verified under ideal conditions that the client can generate traffic and is reachable from the destination server through the router. Next, I tested that traffic can saturate the queue of the router and the host machine can access the router through an FTP session. This was done to validate the accessibility of the queue of the router from the host machine. With the access to the router validation completed, I needed to validate the use of a filter to predict the queue size. This was done in two different ways. The first way was the offline method which collected of all the router queue state files and sent them to a filter function to predict the queue size. The second way was the online method which was the sending of the queue size every prediction time step to a filter through the FTP connection and calculating the prediction of the queue size every ‘x’ seconds. Finally, I verified and compared these results made between MATLAB function and virtual router, and the result calculated by MATLAB only.

3.4 Parameters/Factors/Levels

3.4.1 First Goal: Queue State Predictor Model

The term *parameter* is commonly defined as any feature or setting that can be changed and affects system performance [18]. The following parameters considered in the queue state predictor model:

1. Prediction period – define as the future prediction time of queue state, measured in seconds. Previous work showed that with relatively small prediction time, the

Extended Kalman filter has better prediction performance [4]. However, in this model, two prediction times has been used; 1 second and 5 seconds.

2. Filter – the Extended Kalman filter was the only filter used to predict the future queue state in previous work. However, new filters were designed and used in this model; the Basic Kalman filter, the Extended Kalman filter which was the same as the previous work, and the Unscented Kalman filter.
3. Network Topology – Two network topologies were designed to validate Stuckey's stochastic control scheme prediction performance; 1 router network topology and 2 routers network topology. These two network topologies will be demonstrated in the following subsections.

3.4.1.1 1 Router Network Topology

This is a simple client/server model that had three virtual machines, client, router, and server. As seen in the Figure 3.8 below, the client generated UDP traffic and routed to the destination server through the router. The router executed a script at each prediction time step to save the current queue state to a file. The host machine accessed the router through an FTP connection to get the queue state file and then passed the current queue size and current sampling time to the filter. The goal of this model is to test the connectivity and the validation of all three types of filters (the Basic filter, the Extended Kalman filter, and the Unscented Kalman filter) which were written in MATLAB in the host machine with the RouterOS VM. This validation was done to check the consistency between post processing and real time processing as described before. This model verified the filters work and validated the Stuckey's Extended Kalman filter estimator performance. In addition, this model allowed comparisons of the prediction performance of the three filters under various network traffic types. This was the same network topology that Stuckey's OPNET model of the simple queuing system had used [3].

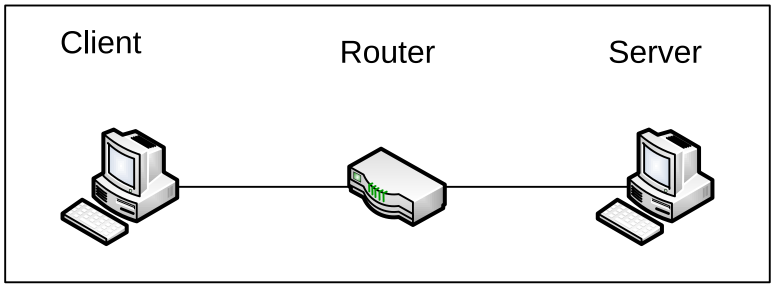


Figure 3.8: 1 router network topology

3.4.1.2 2 Routers Network Topology

This 2 routers topology had four clients, two routers, and two servers. This topology had multiple clients to generate the traffic with two different destination servers as shown in Figure 3.9. At each prediction time step, the host machine accessed each router through an FTP connection to get the current queue state. Each filter then made its future prediction of queue state. This topology was also used in Stuckey's OPNET scenario [3].

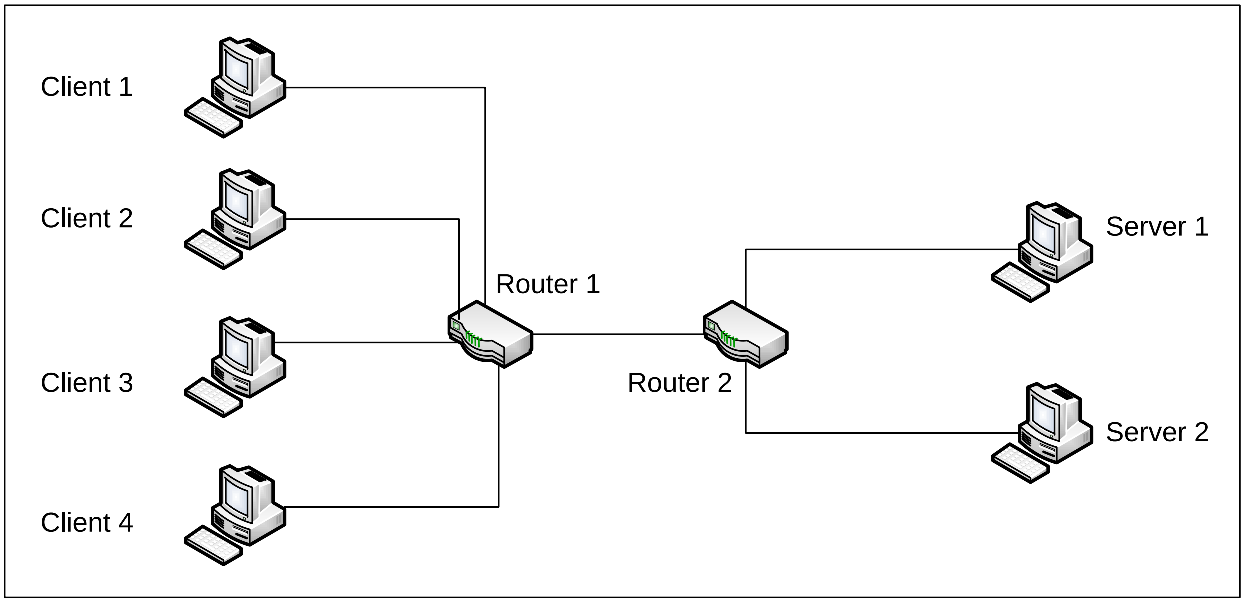
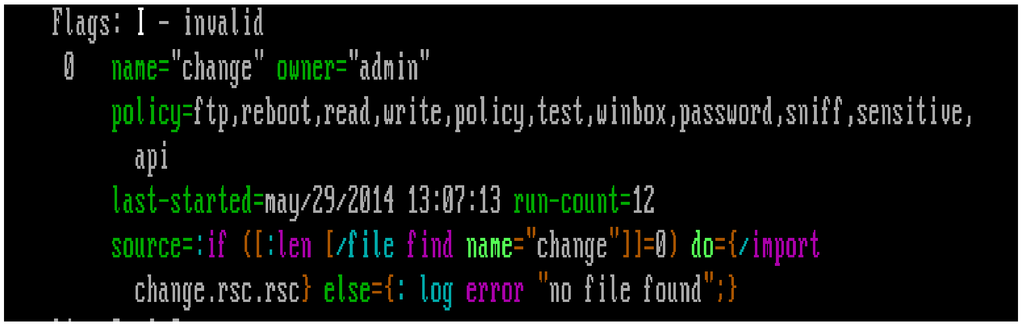


Figure 3.9: 2 router network topology

3.4.2 <u>Second Goal:</u> Queue Controller Model

With filters predictions of the network states done in the previous topologies, we can create an application that can utilize them. This application called DRQC which is implemented by Haught (see Chapter II for more details) [1]. The overall goal of DRQC is to optimize the computer network by reducing the network congestion. It makes a decision at each prediction time step based on the current and predicted network states. These network states are the current and the predicted router queue size.

In this topology, we applied the DRQC in virtual network environment to improve the network performance. As seen from Figure 3.11, four clients generated their own traffic with different paths, four routers, and two destination servers. For instance, client1 communicated server1 through the route client1➔ RouterA ➔ RouterB ➔ RouterD ➔ server1. Also, client3 communicated server1 through the route client3 ➔ RouterB ➔ RouterD ➔ server1. If the prediction of RouterB's queue size was above normal; more than 85% of the actual queue size, based on the queue size prediction received from the filter, DRQC will take an action. This action will allow RouterB to report its queue state to RouterA to change client1's route since it has more traffic priority than client3. RouterA will execute the change route script as shown in figure 3.10 below, so, the new route of client1 will be client1 ➔ RouterA ➔ RouterC ➔ RouterD ➔ server1.



```
Flags: I - invalid
 0   name="change" owner="admin"
     policy=ftp,reboot,read,write,policy,test,winbox,password,sniff,sensitive,
       api
     last-started=may/29/2014 13:07:13 run-count=12
     source=:if ([:len [/file find name="change"]]=0) do={/import
       change.rsc.rsc} else={: log error "no file found";}
```

Figure 3.10: Change route script

By applying DRQC, we prevented RouterB from dropping high priority packets and therefore improving the performance of the network.

In the queue controller model, we considered the same parameters as the first model, queue state prediction model. We used 1 second and 5 seconds future queue prediction period. We applied the three types of filter, the Basic filter, the Extended Kalman filter, and the Unscented Kalman filter with different network scenarios to compare the prediction performance. Also, we used the 4 routers network topology as seen in Figure 3.11 in this model. In addition, we validated which filter gave a better network performance in terms of packets loss.

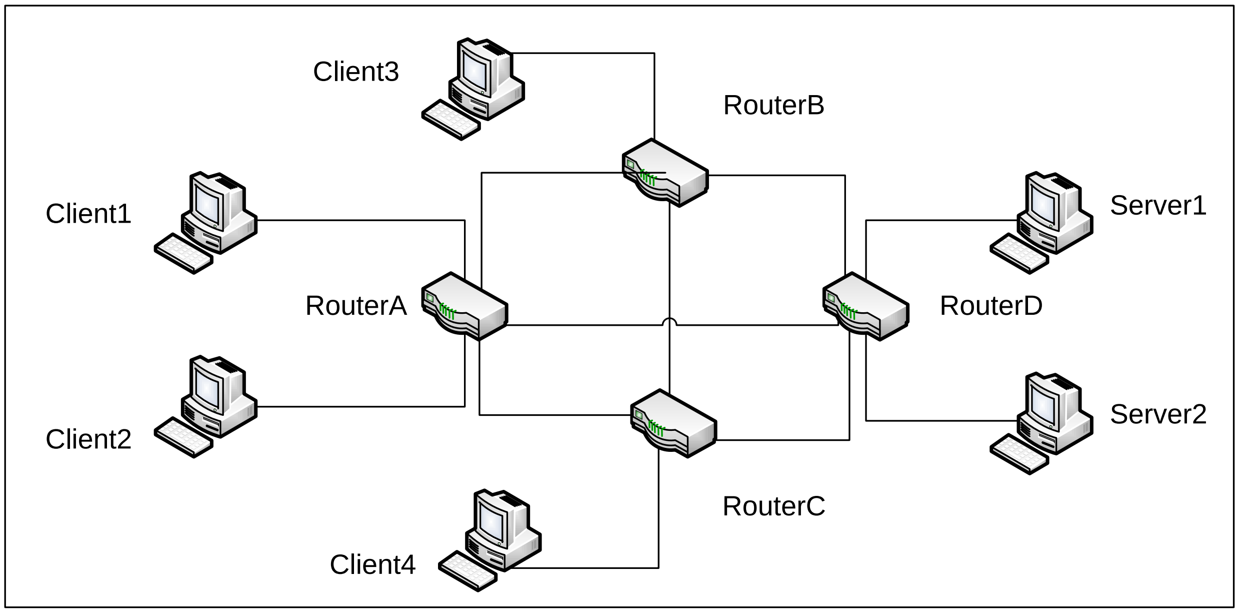


Figure 3.11: 4 routers network topology

3.5 Performance Metrics

The performance metrics measured in this research were:

The actual and predicted RouterOS VM queue size – These measurements supported the validation of the prediction performance of Stuckey's estimator. The actual queue size was measured by accessing the router through an FTP connection. These measurements passed to the filters to measure the predicted queue size. Mean Square Sum Error (MSSE), Mean Absolute Percentage Error (MAPE), and Percent Error of Difference of average actual and predicted queue size were used to compare the filters' prediction performance. Actual and predicted queue size are measured in packets.

Packets dropped due to overwhelming RouterOS VM queue size – The ability to minimize dropping packets improves network performance. Filters predicted the future of queue state and this prediction allowed the network to be adjusted to prevent congestions and dropping packets using DRQC. Packets dropped is measured in packets (1500 bytes per packets).

3.6 System Workloads

In this research, three different topologies were designed. Different traffic load is generated for these different topologies to saturate and predict the RouterOS VM queue size. Each virtual clients transmitted different files to destination virtual server using UDP sockets. Each RouterOS queue had capacity of 250 packets and each packet containing 1500 bytes. The packet arrival rate of each queue was 2 Mbps with constant service rate of 512 Kbps.

3.7 Experimental Design

There were two models designed in this research. Each model had its own network system parameters which were network topology, prediction period, and the number of filter has been used to predict the RouterOS queue size. Each model ran for 15 times. Each ran has different traffic sending time and utilizing different noise associated with the filters. These runs would allow for collecting and analyzing the performance metrics. Table 3.1, shows the total number of experiments conducted for each model. There were 180 experiments conducted for the queue state prediction model and 90 experiments conducted for queue controller model. A total of 270 experiments performed to validate Stuckey's estimator and Haught's DRQC in virtual network environment.

Table 3.1: Test Matrix

	Network Topology	Prediction Period	Filter	run	Total
Queue State Prediction Model	2	2	3	15	180
Queue Controller Model	1	2	3	15	90
Total	-	-	-	-	270

3.8 Expected Results

- Demonstrating the performance of the Extended Kalman filter in virtual network environment to validate Stuckey' extended Kalman filter estimator. Tables and graphs will be presented including the prediction performance of all three filters, the Basic filter, the Extended Kalman filter, and the Unscented Kalman filter.
- The Unscented Kalman filter might give the same or better prediction performance compared with the Extended Kalman filter.
- Demonstrating the performance of the queue controller model in virtual network environment to validate Haught's DRQC.
- Applying DRQC might improve the performance of the network in terms of packet dropped and end to end delay.

3.9 Summary

In conclusion, we started with an introduction and overall goal of this research. The overall goal is to validate the performance of the Extended Kalman filter estimator developed by Stuckey against a new filters estimators and models and applying DRQC developed by Haught in a virtual network environment. In these models, we implemented a new controller and estimator using the Unscented Kalman filter. Also, we chose UDP traffic, interarrival rate, service rate, and prediction time for each scenario to validate Stuckey's stochastic theory and Haught DRQC. We designed two models, queue state predictor model and queue controller model. The performances of these designs are demonstrated in chapter 4.

4. Results and Analysis

This chapter presents the results of the two models, the queue state prediction model and the queue controller model that were presented in Chapter 3. The first section of this chapter details the results and analysis of the queue prediction model to describe how well the Kalman filters (EKF and UKF) perform in a virtual network. The second section presents the results of the queue controller model and shows how the queue controller improves the network performance in a virtual network environment.

4.1 Results of the Queue State Prediction Model

In previous research, the Mean Absolute Percent Error (MAPE) and percent error of difference of averages of actual and predicted queue size were used to describe the performance of the extended Kalman filter [3,4]. MAPE can be described as following:

$$MAPE = \frac{100}{n} \sum_{t=1}^{n} \left| \frac{A_t - P_t}{A_t} \right| \qquad (2.18)\ [30]$$

Where A_t and P_t are the actual queue size and the predicted queue size, respectively at time t, and n is the total time. As you can see from Equation 2.18 above, if the actual queue size is zero at time t, there will be indeterminate form due to divide by zero. Therefore, this method fails whenever the actual queue size is zero and it does not provide a precise result of the prediction error and its effect on the prediction performance of the filter.

The other method was calculating the percentage error of the difference between the actual and predicted queue size. This method can be describe as follows:

$$Percent\ Error\ of\ Diffrence = \left| \frac{Average\ Actual\ Queue\ Size - Average\ Predicted\ Queue\ Size}{Average\ Actual\ Queue\ Size} \right| *100 \qquad (2.19)$$

The problem with this method is that, it does not consider the filter prediction at every time step. The filter may predict the queue size so badly in some situations during the simulation due to the amount of traffic in the queue.

To overcome these problems with MAPE and Percent Error of Difference, the Mean Sum Square Error (MSSE) is used as the primary method to evaluate prediction performance. This method can be described as follows:

$$MSSE = \frac{\sum_{t=1}^{n}(A_t - P_t)^2}{n} \qquad (2.20)$$

Where A_t is the actual size of the queue at time t, P_t is the predicted queue size at time t, and n is the total time.

For determining how well the filters predict the queue size, these three methods were used to analyze the filters' performance of these filters and their relative goodness are shown in Figure 4.1.

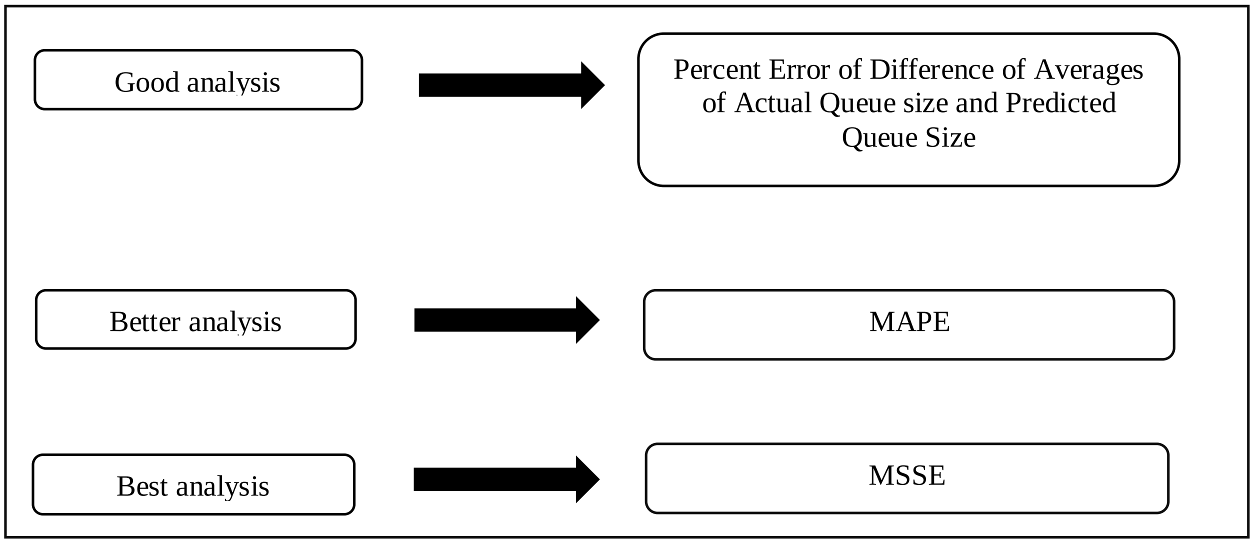

Good analysis
Percent Error of Difference of Averages of Actual Queue size and Predicted Queue Size
Better analysis
MAPE
Best analysis
MSSE

Figure 4.1: Prediction Performance Evaluation methods

As shown in Figure 4.2, for the queue state prediction model, there were two topologies, 1 router network topology and 2 routers network topology. For each topology, two prediction periods were set for each filter to make queue size future state prediction, a 1 second period and 5 second period. The first set of results details the simulations using 1 router network topology with 1 second and 5 seconds prediction periods. The second set of results present the simulation results using 2 routers network topology with 1 second and 5 second period prediction periods. Finally, these results present the prediction performance of the three filters, the basic filter, the extended Kalman filter, and the unscented Kalman filter for both topologies and prediction periods.

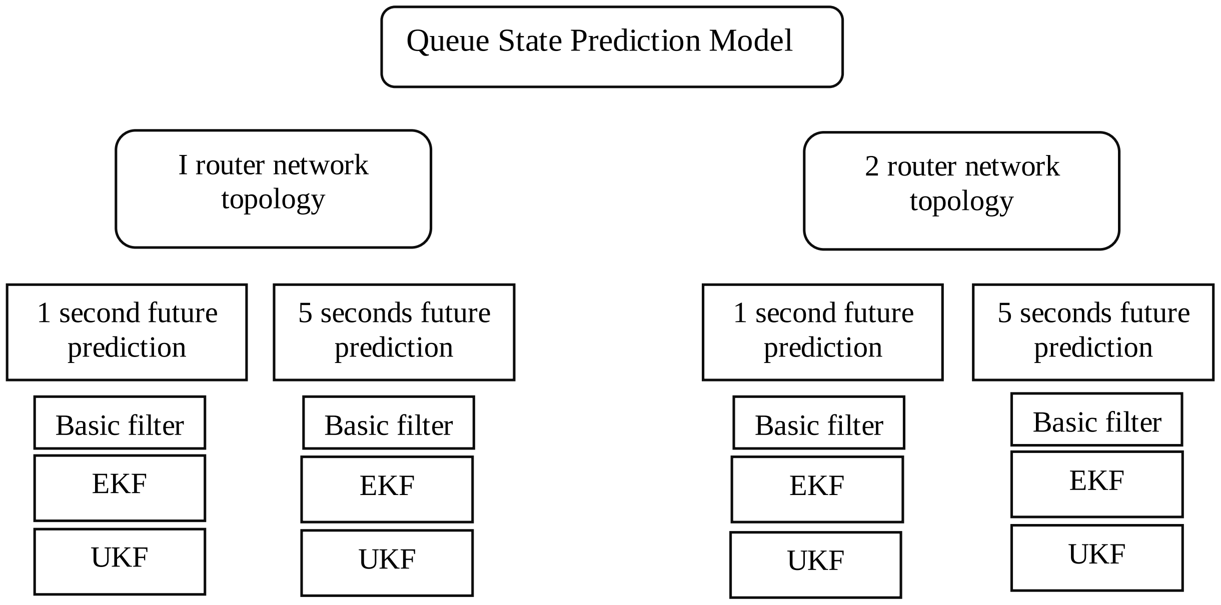

Queue State Prediction Model
I router network topology
2 router network topology
1 second future prediction
5 seconds future prediction
1 second future prediction
5 seconds future prediction
Basic filter
EKF
UKF
Basic filter
EKF
UKF
Basic filter
EKF
UKF
Basic filter
EKF
UKF

Figure 4.2: Result of Queue State Prediction Model chart

4.1.1 1 Router Network Topology & 1 Second Period Time

The following results describe the prediction performance of the three filters, the basic filter, the extended Kalman filter, and unscented Kalman filter. We used 1 router network topology described in Section 3.4.1.1 and set the prediction period to 1 second.

Table 4.1: Results of 1 router network topology with 1 second prediction period

	Basic Filter	EKF	UKF
Actual Queue Size Average	160.0756	160.0756	160.0756
Predicted Queue Size Average	153.5976	160.9546	154.0385
Difference of Averages	6.478	0.878967	6.037091
Percent Error	4.046838 %	0.549095 %	3.7714 %
MAPE	11.66812 %	7.518123%	7.5481 %
MSSE	550.2891	250.2608	248.5673

Table 4.1 shows the results from an average of 15 run simulations and shows the results of the three filters prediction using 1 router network topology with a 1 second prediction period. Each separate simulation has different traffic sending time and utilizing different noise associated with the filters. The basic filter has the largest percent error of difference of average actual queue size and average predicted queue size, MAPE, and MSSE. The basic filter was designed as a base line prediction filter to be used as a reference to evaluate the EKF and the UKF prediction performance. In MAPE and percent error of difference, the EKF showed the best performance. However, as we discussed earlier, the primary method for evaluating the performance of prediction is MSSE. The UKF has the smallest error of MSSE and therefore gave the best prediction performance compared with other filters. It is 0.681% better than the EKF.

The following graph shows the average actual queue size and the average predicted queue size for the three filters across 15 run simulations.

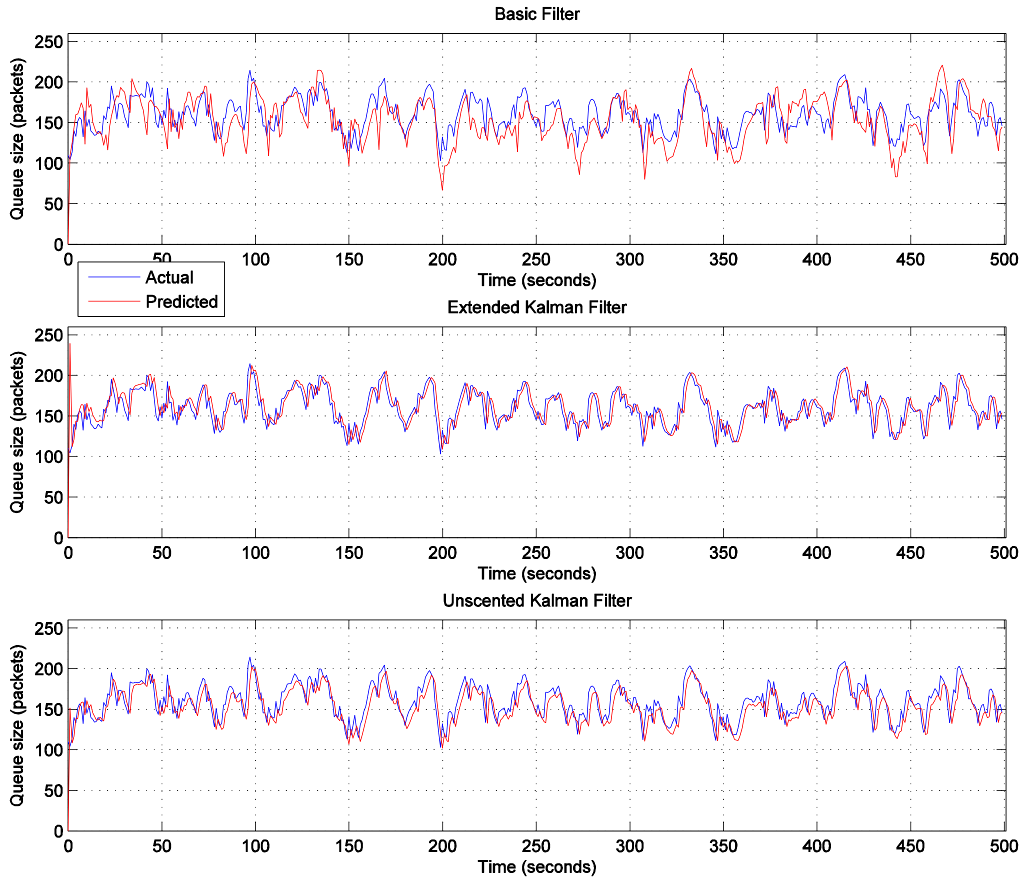


Figure 4.3: Prediction of actual queue size using 1 router network topology & 1 second prediction period time for all filters, the basic filter, the EKF, and the UKF. Blue line: actual queue size, red line: predicted queue size.

These graphs show the results of prediction performance of the three filters, the basic filter, the EKF, and the UKF. The blue line is the actual queue size and the red line is the predicted queue size. From naked eye, it is obvious the basic filter gives the worst prediction performance compared with the other filters. The EKF and UKF give similar prediction performance. However, the UKF predicted the queue size better than the EKF according to MSSE in Table 4.1.

The following graph show the raw error between the average of queue size and the average predicted queue size for all three filters.

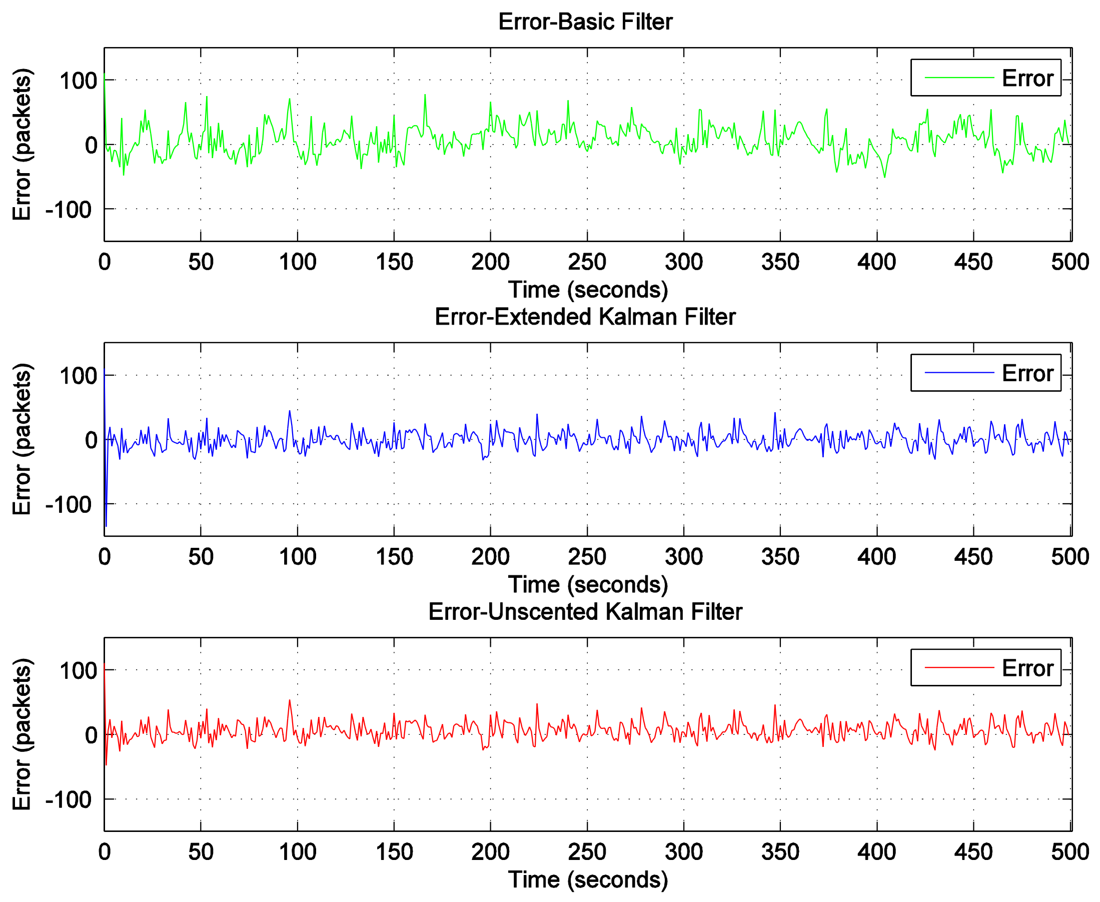


Figure 4.4: Error of difference of actual queue size and predicted queue size for all filters using 1 router network topology & 1 second prediction period time.

Figure 4.4 shows the error between the actual queue size and predicted queue size at every second. As described in Table 4.1, the basic filter has the largest error compared with other filters.

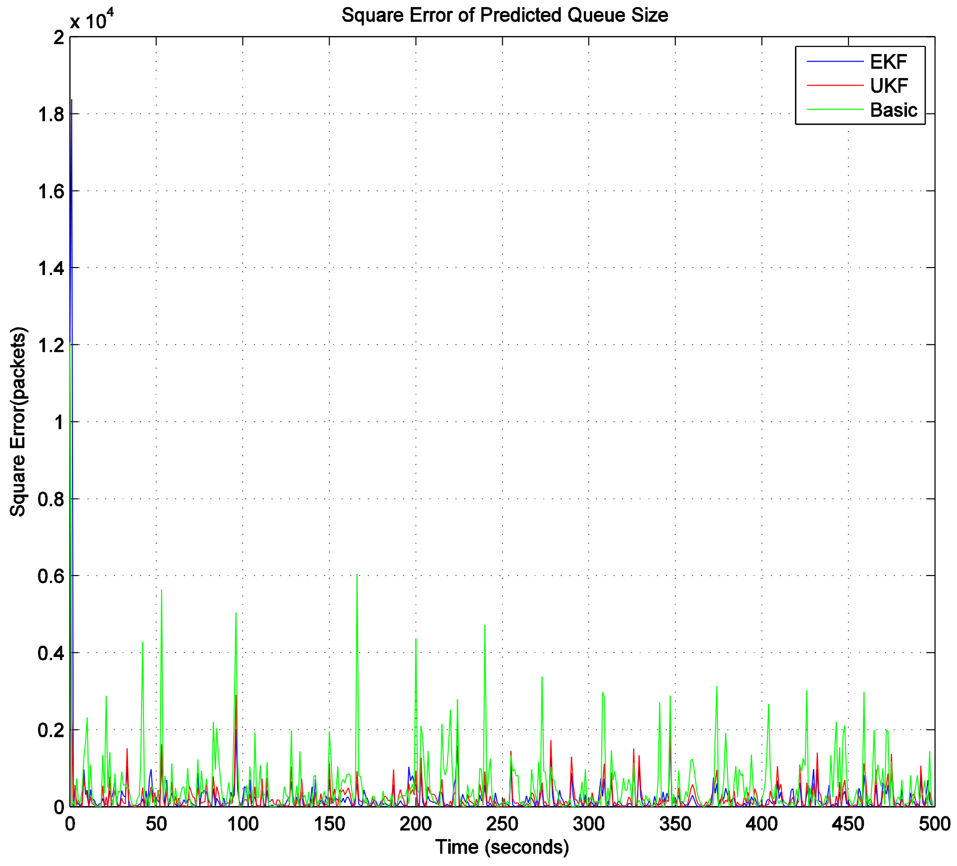


Figure 4.5: Square error of predicted queue size of all three filters using 1 router network topology & 1 second prediction period time

Figure 4.5 shows the average square error of three filters. The green line represents the square error of the basic filter, the red line is the square error of the UKF and the blue line is the square error of the EKF. The UKF has the smallest average sum square error compared with other filters and the basic filter has the largest average sum square error as described in Table 4.1.

4.1.2 1 Router Network Topology & 5 Seconds Period Time

This section presents the results of filter prediction performance using the 1 router network topology and 5 second prediction period.

Table 4.2: Results of 1 router network topology with 5 second prediction period

	Basic Filter	EKF	UKF
Actual Queue Size Average	159.7832	159.7832	159.7832
Predicted Queue Size Average	160.8723	165.4855	162.9478
Difference of Averages	1.089109	5.702294	3.16462
Percent Error	0.681617 %	3.56877 %	1.980572 %
MAPE	21.74028 %	11.71432 %	9.535767 %
MSSE	1577.969	800.739	359.0406

Table 4.2 shows the results of filter prediction performance from an average of 15 simulation runs. As you can see, more error was generated between actual queue size and predicted queue size if the filters were set to predict a greater future time because of lack of samples at each prediction time and there is no indication the traffic changing rapidly in the future. The fewer samples collected and passed to the filter affects the filter prediction performance. From the result above, the basic filter has the smallest percent error of difference of average actual queue size and average predicted queue size. In MAPE, the UKF has the smallest error compared with other filters. However, the UKF has the best prediction performance because it has the smallest MSSE. It is 123.02% and 339.49% better than the EKF and the basic filter respectively.

The following graph shows the average of actual queue size and predicted queue size for the three filters for an average across 15 simulation runs.

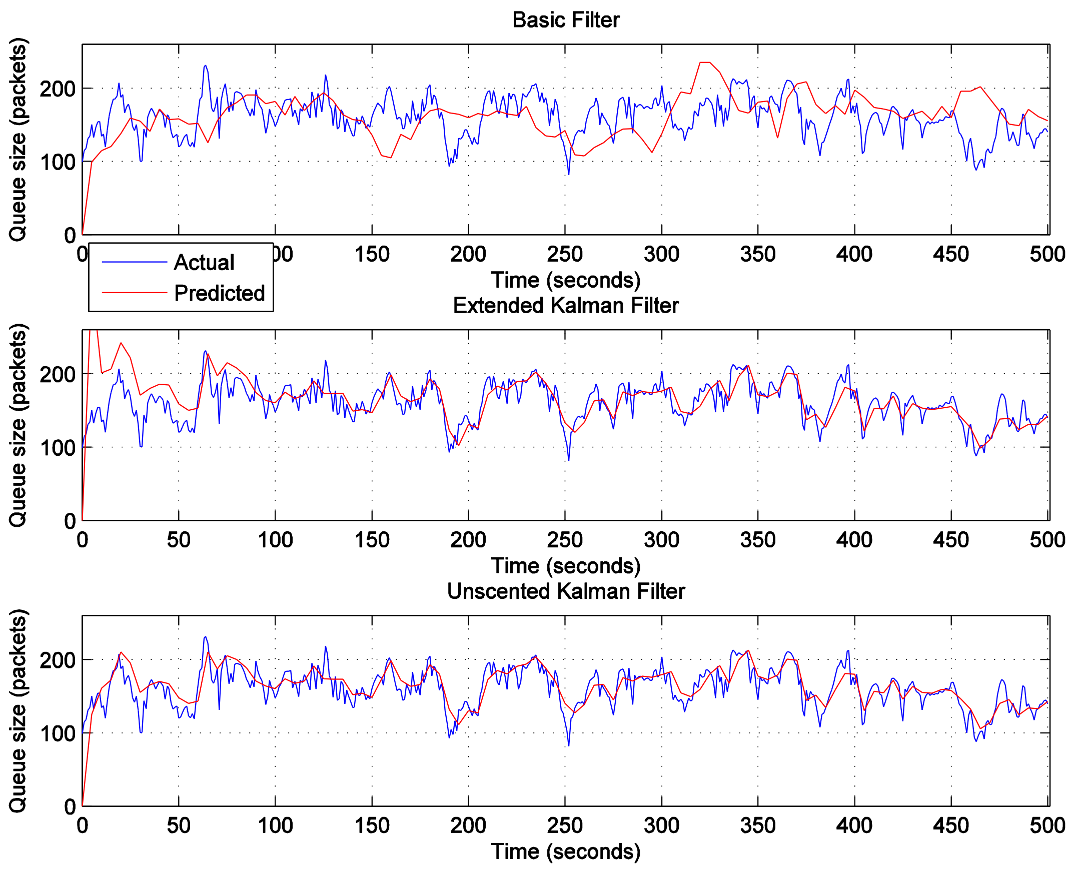


Figure 4.6: Prediction of actual queue size using 1 router network topology & 5 second prediction period time for all filters, the basic filter, the EKF, and the UKF. Blue line: actual queue size, red line: predicted queue size

Figure 4.6 shows the prediction performance of the three filters. Blue line represents the actual queue size and the red line represents the predicted queue size. As you can see, The UKF gives the best prediction performance and the basic filter gives the worst prediction performance. The EKF prediction follows the trend of actual queue size but not accurate as the 1 second future time prediction in the previous section.

Figure 4.7 and Figure 4.8 show the error between the actual queue size and the predicted queue size and the square error of the three filters respectively.

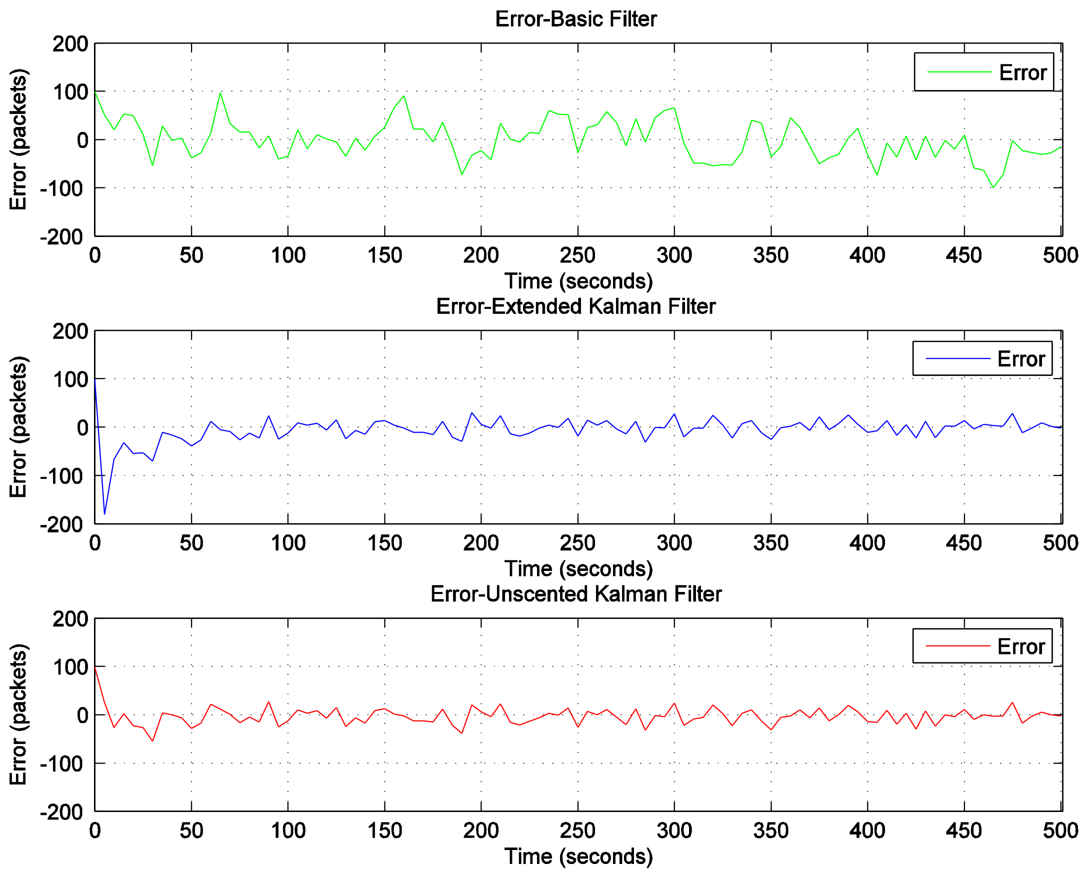


Figure 4.7: Error of difference of actual queue size and predicted queue size for all filters using 1 router network topology & 5 second prediction period time.

Figure 4.7 shows the error between the actual queue size and predicted queue size at every second. The basic filter has the largest prediction error compared with other filters. By keeping in mind, 5 seconds future time prediction is too long to make a good prediction and lack of samples, the UKF gives the smaller prediction error.

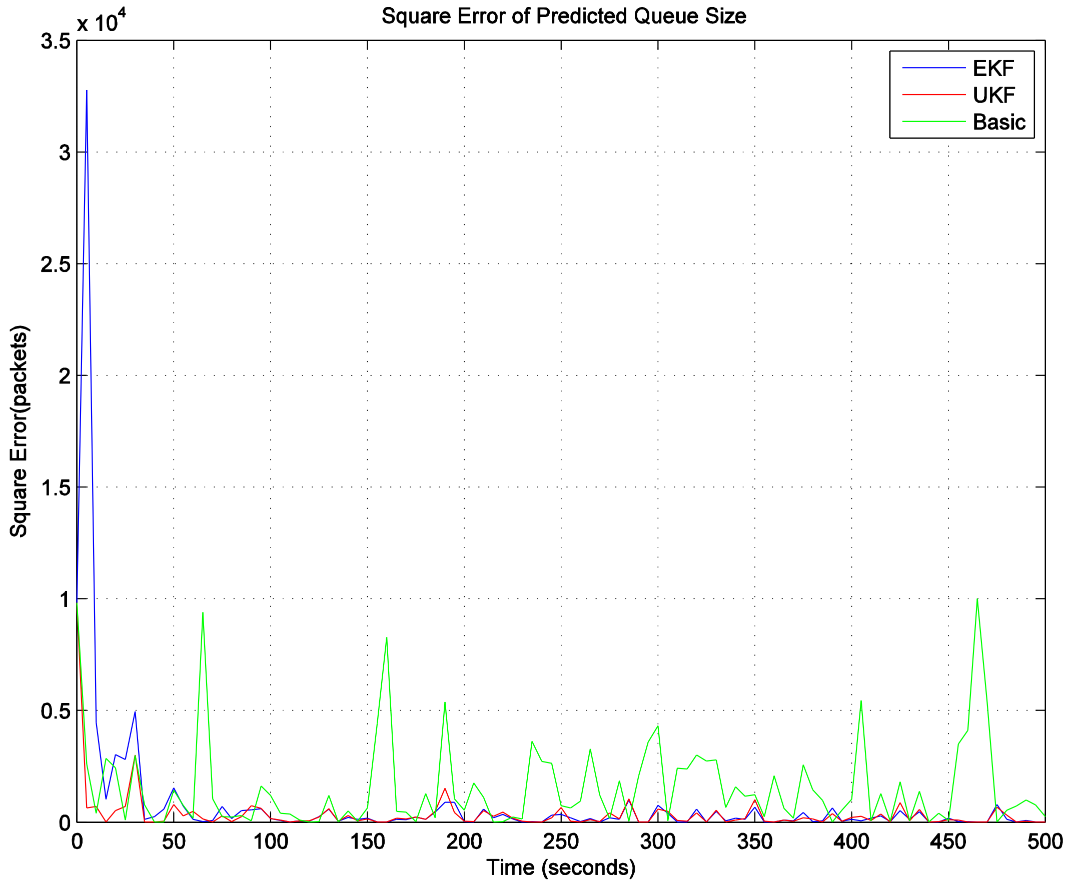


Figure 4.8: Square error of predicted queue size of all three filters using 1 router network topology & 5 second prediction period time.

Figure 4.8 shows the average square error of three filters. The green line represents the square error of the basic filter, the red line is the square error of the UKF and the blue line is the square error of the EKF. The UKF has the smallest average sum square error compared with other filters and the basic filter has the largest average sum square error.

To sum up, the UKF gave the best prediction performance with 1 and 5 second prediction periods for 1 router network topology. The next section, will present the results of filter prediction performance using 2 routers network topology and the two prediction periods.

4.1.3 2 Routers Network Topology & 1 Second Period Time

Using 2 routers network topology as described in section 3.4.1.2, allows to generate more traffic from different nodes and provide a better evaluation of filter prediction performance. This section contains the results of prediction performance of the three filters when each filter was set to predict the queue size 1 second ahead in the future.

Table 4.3: Results of 2 router network topology with 1 second prediction period

	Basic Filter	EKF	UKF
Actual Queue Size Average	133.5366	133.5366	133.5366
Predicted Queue Size Average	134.8828	134.6335	128.1442
Difference of Averages	1.3462	1.096941	5.392446
Percent Error	1.008113 %	0.821453 %	4.038179 %
MAPE	17.44411	8.918679 %	9.474936
MSSE	753.515	276.5229	268.5832

Table 4.3 shows the prediction performance of the three filters from the average of 15 simulation runs. The EKF, has the smallest error in percent error of difference of averages and MAPE. In percent error of difference of averages, the basic filter has the smaller prediction error than the UKF. However, In MSSE which is the best evaluation method of prediction performance, the UKF is 2.9561% and 108.5517% better than the EKF and the basic filter respectively. Therefore, the UKF, has the best prediction performance of queue size.

The following graph shows the average of actual queue size and predicted queue size for the three filters across 15 simulation runs.

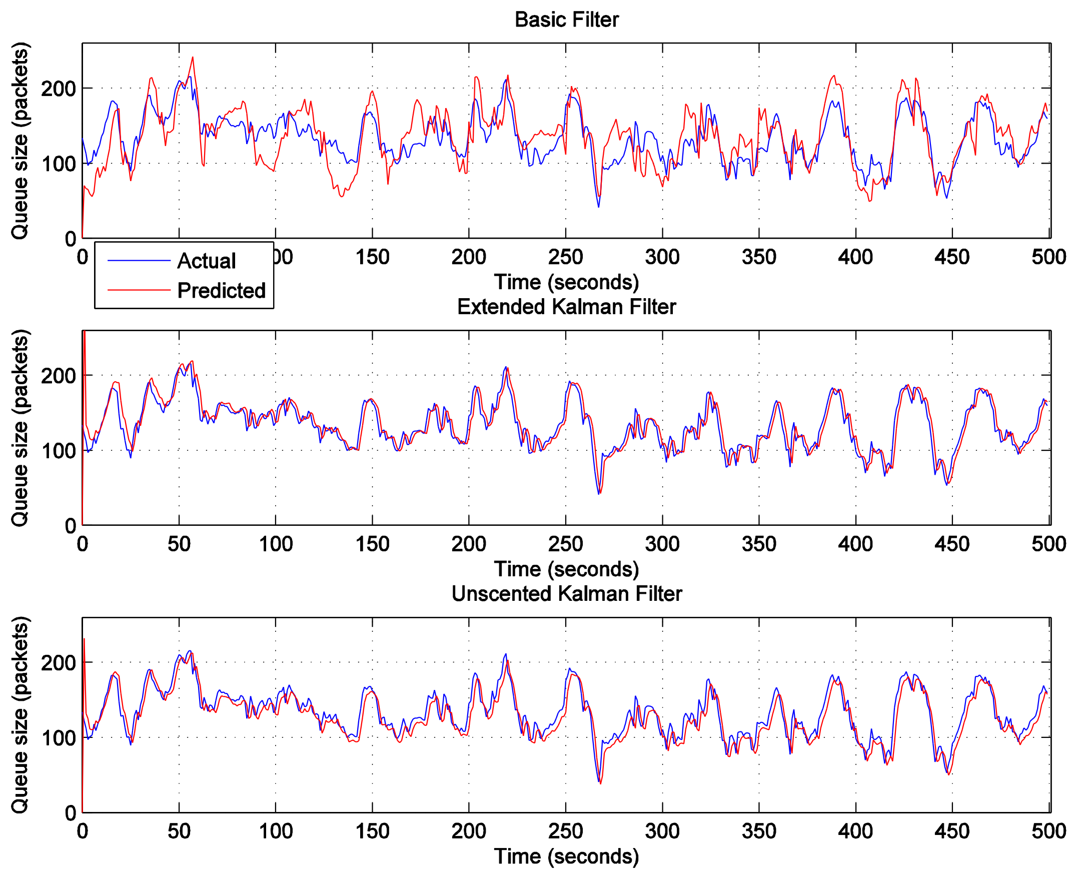


Figure 4.9: Prediction of actual queue size using 2 router network topology & 1 second prediction period time for all filters, the basic filter, the EKF, and the UKF. Blue line: actual queue size, red line: predicted queue size

Figure 4.9 shows the prediction performance of the three filters. Blue line represents the actual queue size and the red line represents the predicted queue size. From the graph, the basic filter gives the worst prediction performance compared with the other filters. The EKF and UKF give similar prediction performance.

Figure 4.10 and Figure 4.11 shows the error between the actual queue size and the predicted queue size and the square error of the three filters respectively.

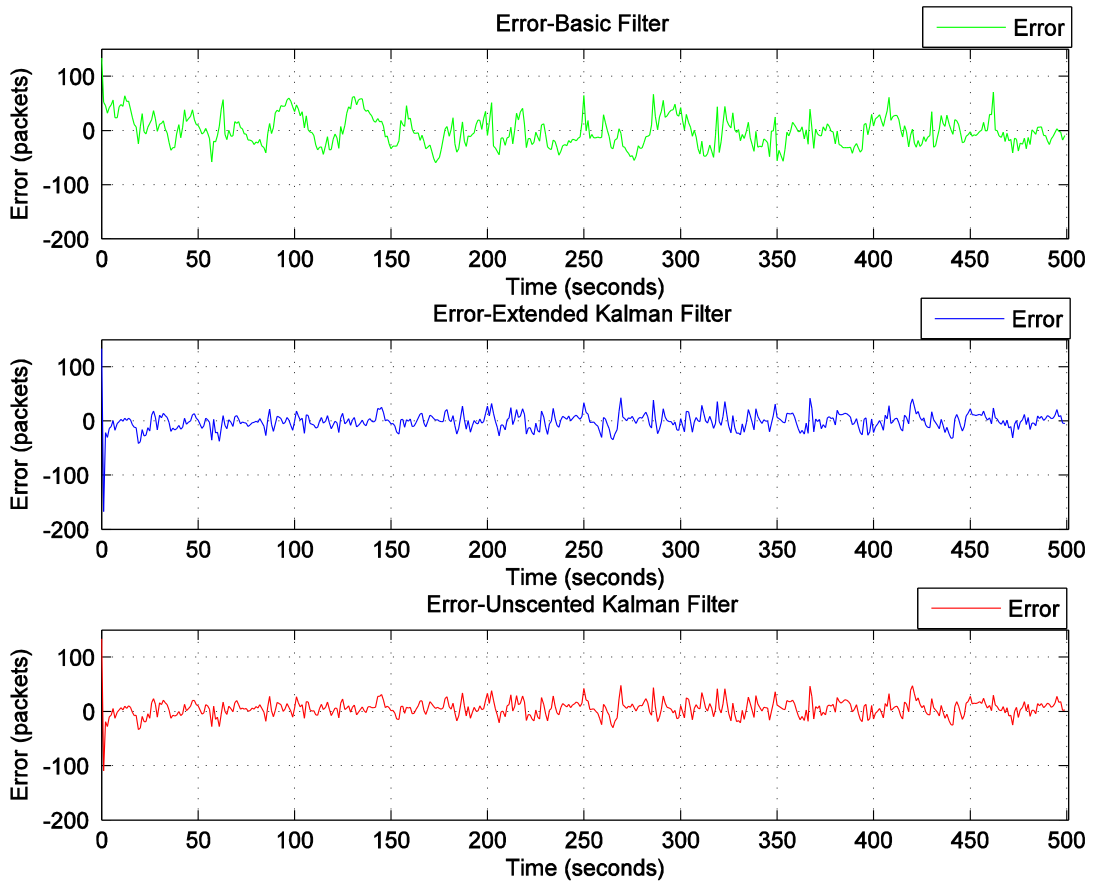


Figure 4.10: Error of difference of actual queue size and predicted queue size for all filters size using 2 router network topology & 1 second prediction period time.

Figure 4.10 shows the error between the actual queue size and predicted queue size at every second. As described in Table 4.3, the basic filter has the largest error compared with other filters.

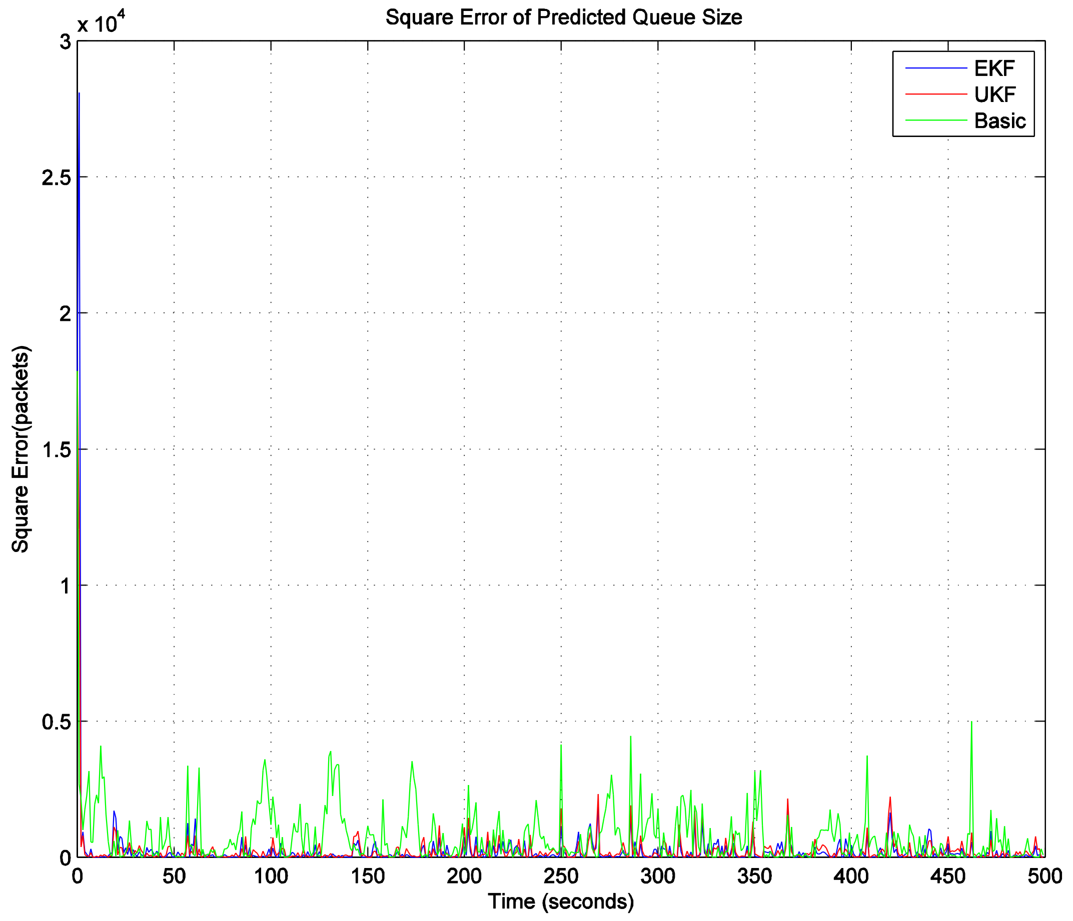


Figure 4.11: Square error of predicted queue size of all three filters size using 2 router network topology & 1 second prediction period time.

Figure 4.11 shows the average square error of three filters. The green line represents the square error of the basic filter, the red line is the square error of the UKF and the blue line is the square error of the EKF. The UKF has the smallest average sum square error compared with other filters and the basic filter has the largest average sum square error.

4.1.4 2 Router Network Topology & 5 Seconds Period Time

This section contains the results of prediction performance of the three filters using 2 routers network topology as described in section 3.4.1.2 and each filter was set to predict the queue size 5 seconds ahead in the future.

Table 4.4: Results of 2 router network topology with 5 second prediction future time

	Basic Filter	EKF	UKF
Actual Queue Size Average	133.595	133.595	133.595
Predicted Queue Size Average	144.2129	135.3336	132.2345
Difference of Averages	10.61782	1.738517	1.360574
Percent Error	7.947766 %	1.301334 %	1.018432 %
MAPE	28.26828 %	12.9446 %	11.35348 %
MSSE	1575.541	684.9227	396.2988

Table 4.4 shows the prediction performance of the three filters from 15 simulation runs. The UKF has the smallest percent error of difference of averages, MAPE, and MSSE compared with other filters. Therefore, the UKF, has the best prediction performance of queue size. The following graph shows the average of actual queue size and predicted queue size for the three filters.

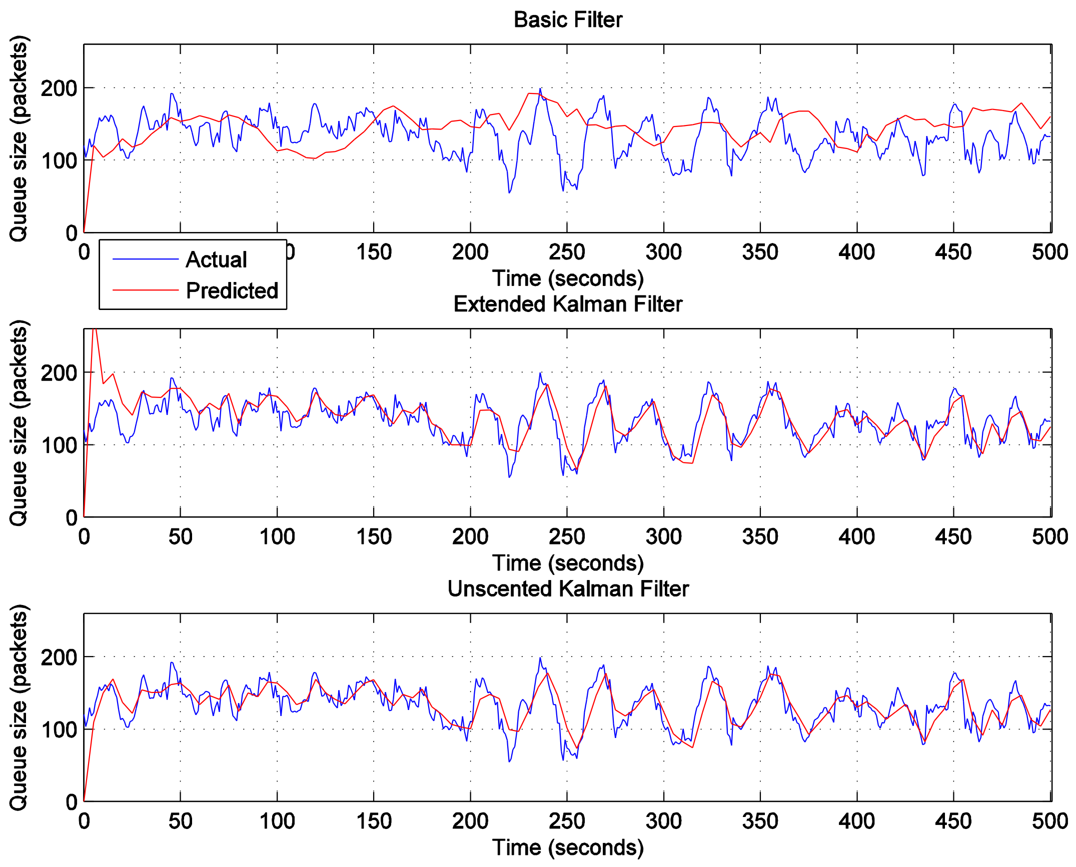


Figure 4.12: Prediction of actual queue size using 2 router network topology & 5 second prediction period time for all filters, the basic filter, the EKF, and the UKF. Blue line: actual queue size, red line: predicted queue size.

Figure 4.12 shows the prediction performance of the three filters. Blue line represents the actual queue size and the red line represents the predicted queue size.

Figure 4.13 and Figure 4.14 show the error between the actual queue size and the predicted queue size and the square error of the three filters respectively.

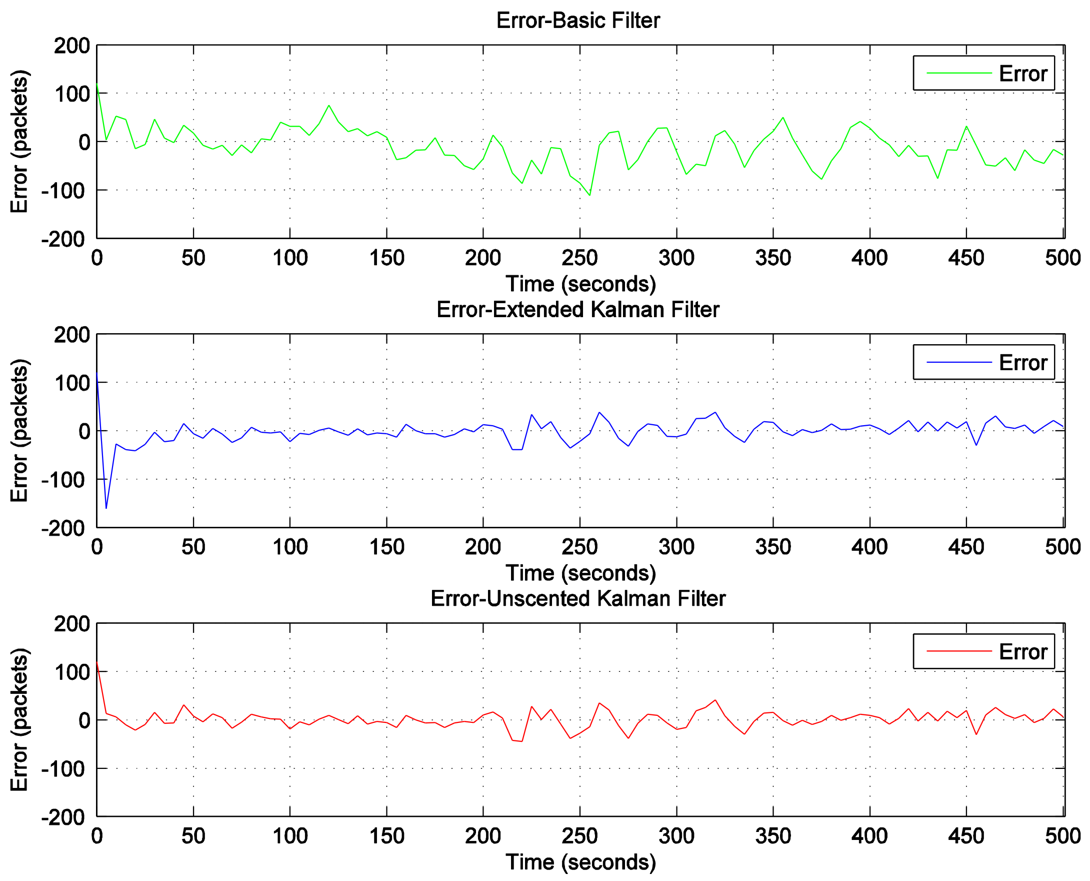


Figure 4.13: Error of difference of actual queue size and predicted queue size for all filters using 2 router network topology & 5 second prediction period time.

Figure 4.13 shows the error between the actual queue size and predicted queue size at every second. The basic filter has the largest prediction error compared with other filters. With 5 seconds prediction future time, even with different topology and more nodes sending traffic, the UKF beats the EKF and gives the smaller prediction error.

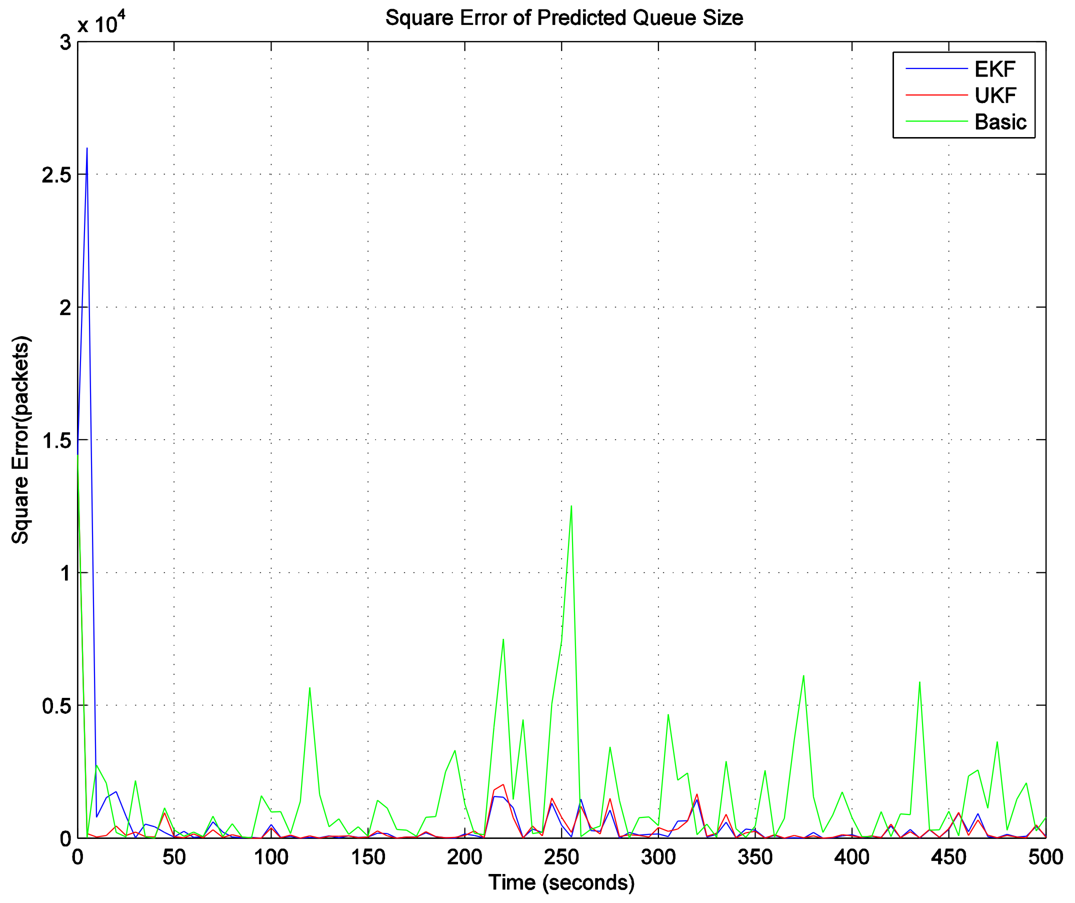


Figure 4.14: Square error of predicted queue size of all three filters using 2 router network topology & 5 second prediction period time.

Figure 4.14 shows the average square error of three filters. The green line represents the square error of the basic filter, the red line is the square error of the UKF and the blue line is the square error of the EKF. The UKF has the smallest average sum square error compared with other filters and the basic filter has the largest average sum square error.

To conclude the results of the queue state prediction model, the UKF gave the best prediction performance for both topologies, the 1 router network topology and the 2 router network topology and for both prediction periods, 1 second prediction period and 5 second prediction periods. This is what we expected in Chapter 3 because the UKF was designed to provide prediction as better as the EKF. However, the worst filter prediction performance for both topologies and prediction periods was the basic filter. As the prediction future time increase to 5 second more error will be generated between the actual queue size and the predicted queue size. This seems reasonable since the amount of traffic inside the queue is changing dramatically and a few samples passed to the filter which made it difficult to predict the queue size accurately. Table 4.5, summarized the best filter prediction performance for both topologies and prediction periods.

Table 4.5: The best filter prediction performance in Queue State Prediction model

Topology	*Prediction Time*	*The Basic Filter*	*The EKF*	*The UKF*
1 router network topology	1 second prediction			✖
	5 seconds prediction			✖
2 routers network topology	1 second prediction			✖
	5 seconds prediction			✖

4.2 Results of the Queue Controller Model

The main goal of the queue controller model was to prevent the router from getting congested and improve the network performance. To do this, the queue controller used the filter queue size prediction methods presented in the previous section. Given the ability to predict the future queue size, the queue controller reduced the number of packet dropped. To change the routing tables in this virtual environment, a script inside the router would run and reroute the traffic based on the current and predicted queue size and the priority of client. This script was described in Figure 3.10 in Chapter 3.

To demonstrate the function of the queue controller, 4 routers network topology was used as described in section 3.4.2. This topology contained four routers (A, B, C, D), four clients (C1, C2, C3, C4), and one server (S1) as shown in Figure 4.15. The purpose of this experiment was to show how the queue controller performed and rerouted the traffic. Table 4.6 shows the default routes of each client along with their priority.

Table 4.6: Clients default routes and their priority

	Priority	Route
Client1 (C1)	2	C1→ A→ B→S1
Client2 (C2)	1	C2→ B→ S1
Client3 (C3)	3	C3→ A→ B→ S1
Client4 (C4)	4	C4→ C→ D→S1

As you can see from Figure 4.15 below, C1 will route by shortest route to the server S1. However, when C2 and C3 start generating its traffic, packet congestion in router B will increase. In order to prevent C2's and C3's packets from being dropped, C1 and C3 will reroute their packets to the new route A→ C→ D→ S1 because C1 and C3 have lower priority than C2.

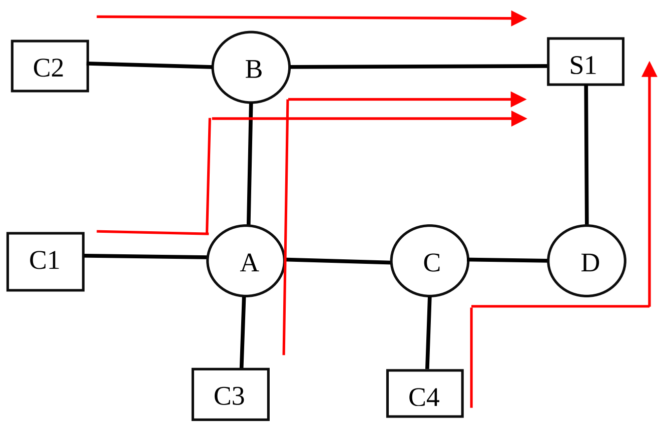


Figure 4.15: Network topology for the queue controller

Again, this experiment was conducted to show the capability of the queue controller to reroute the traffic before the queue size of router B fills up and starts dropping packets.

This experiment has the two scenarios explained below:

1. Without utilization of the queue controller: This scenario shows the network congestion in router B without utilizing the queue controller which results in overwhelming congestion and the dropping of packets.

2. With utilization of the queue controller: This scenario shows how the queue controller performed and made decisions. In this scenario, the queue size of router B will be predicted at 1 second and 5 second intervals. Also, the three filters, the basic filter, the EKF, and the UKF will be used to predict the queue size.

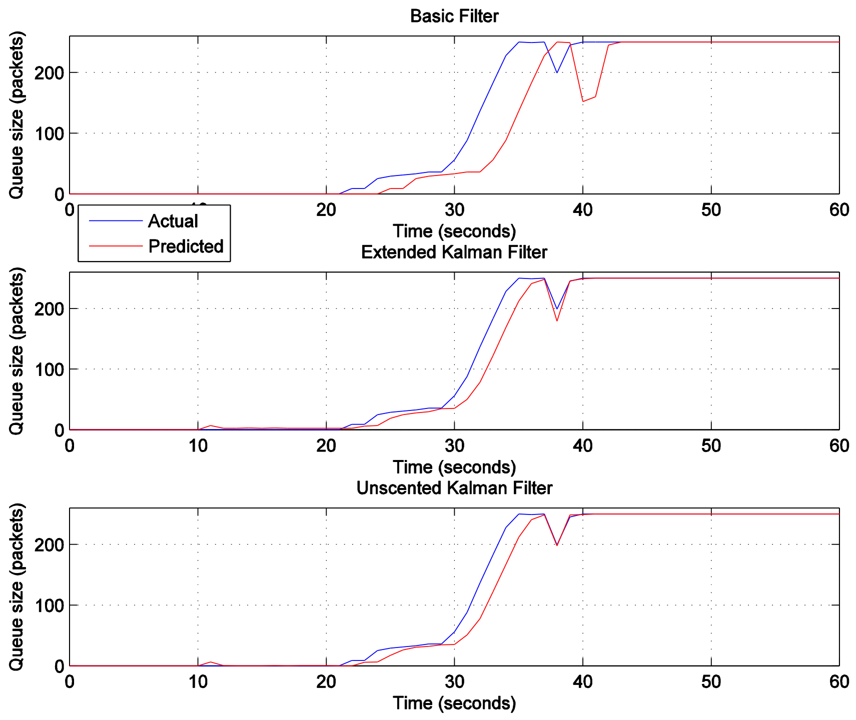


Figure 4.16: Actual queue vs predicted queue size of router B with 1 second prediction period and without using the queue controller

Figure 4.16 above shows the simulation results of router B's queue size when the queue controller was not utilized. The queue size reached its maximum capacity and started dropping packets. Also, it shows, the prediction performance of three filters with a 1 second prediction period.

The result of the queue size of router B when the queue controller was utilized with a 1 second prediction period is shown in Figure 4.17 below. As you can see, when the queue size reached its maximum capacity the traffic rerouted according to the queue controller decision. In addition, the prediction performance with 1 second prediction period of three filters were shown. The UKF gave the best prediction performance of the queue size compared with other filters.

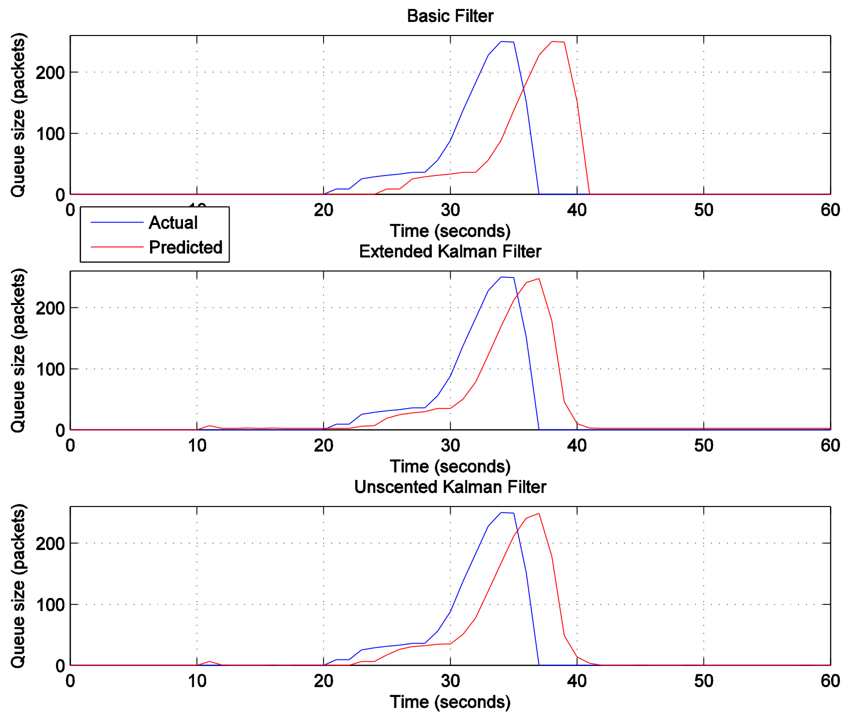


Figure 4.17: Actual queue vs predicted queue size of router B with 1 second prediction period and using the queue controller.

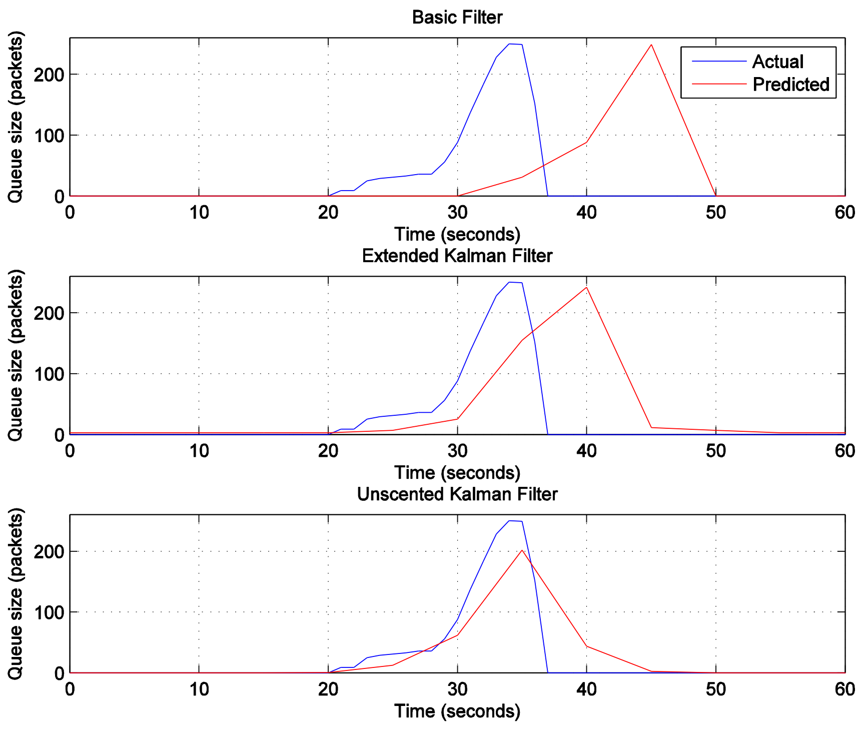


Figure 4.18: Actual queue vs predicted queue size of router B with 5 second prediction period and using the queue controller

Figure 4.18 shows the result of the queue size of router B's when the queue controller used a 5 second prediction period. As you can see, when the queue size reached its maximum capacity the traffic rerouted according to the queue controller decision. In addition, the prediction performance with 5 seconds prediction period of three filters were shown. The UKF gave the best prediction performance of the queue size compared with other filters.

At each prediction step, the host machine was able to access router B through an FTP session to get the current queue state file. From this file, the number of packets dropped was recorded. Figure 4.20 shows the total number of packets dropped for each scenario. In the first scenario, there were 2207 packets dropped when the queue controller was not utilized. In the second scenario, when the queue controller was utilized and with 1 second future time prediction and used the UKF, there were 86 packets dropped. When the EKF and the basic filter were used there were 89 and 623 packets dropped respectively. For the second scenario, with a 5 second prediction period, there were more packets were dropped because the accuracy of the filter prediction degraded due to the longer queue size sample-prediction interval. There were 241 packets dropped when the UKF was used. When the EKF and the basic filter were used there were 360 and 1109 packets dropped respectively. These statistics show that, when the queue controller was used the number of packets dropped significantly reduced and therefore improved the performance of network by reducing congestion. However, with a smaller prediction period, the queue controller performed better. With a 1 second prediction and using the UKF, the number of packets dropped was reduced by 96.10% whereas by using the EKF, it was reduced by 95.59%. By using the UKF with a 5 second prediction period, the total number of packets dropped was reduced by 89.08 % and when the EKF was used, it was reduced further by 83.6 %. With a 5 second prediction interval, more packets were dropped because the filters did not predict the queue size as accurately as with a 1 second prediction period. This was also evident in the previous results of the queue size state prediction model. In addition, in the queue state prediction model results, the basic filter was the worst predictor compared with other filters and hence there were more packets dropped because it predicted the queue size worst of all, compared to the EKF and the UKF.

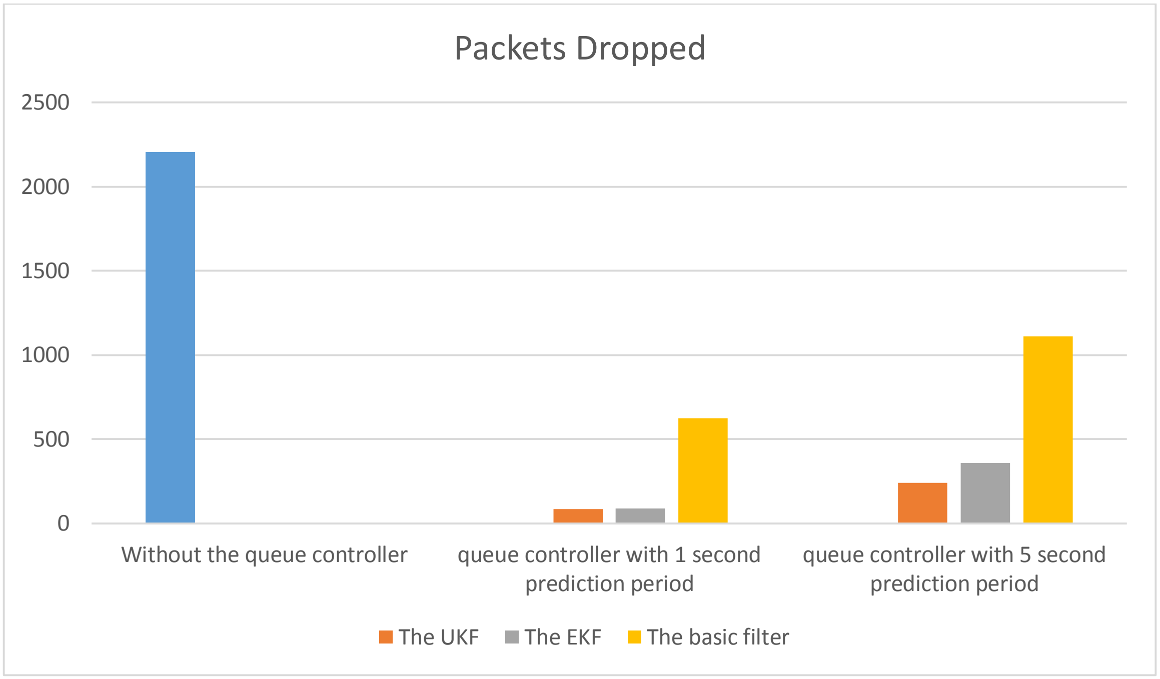


Figure 4.19: Dropped packets at router B in the queue controller simulation

In this chapter, two models were designed, the queue size state prediction model and the queue controller model. In the first model, queue size was predicted by using three types of filters, the basic filter, the EKF, and the UKF with two different prediction periods, 1 and 5 second. The UKF gave the best prediction performance compared with other filters. In the second model, the queue controller used the filter queue size state prediction model to reduce the number of packets dropped and therefore improved the network performance.

5. Conclusion

Today, modern congestion and routing management algorithms may work well for networks with static topologies and moderate QoS prerequisites. However, these algorithms may not suitable for modern military networks and other sensitive organizational communications because these networks take place in an open environment that include mobile networks which have many assets, numerous elements and dynamic network topologies. Also, they need to be secured, require a high level of Quality of Service (QoS), and able to adapt to changing demands to prevent from service interruptions. These networks may able to be flexible to accomplish mission goals through the utilization of queue size prediction.

This chapter addresses the objectives and the conclusion of this research. Section 5.1 summarizes the scope and the objectives of this research. Section 5.2 presents the research contributions. Section 5.3 presents the possibility and recommendations of future research.

5.1 Research Scope and Objectives

The idea of the network weatherman implemented by Stuckey was to predict the router queue size. He designed network models to predict the queue size through the utilization of the extended Kalman filter simulated in an OPNET simulator. Stuckey’s work was extended by James Haught, who tried to validate the efficiency of the extended Kalman filter prediction of the router queue size using NS2 network simulation. He performed several simulation scenarios using NS2 which consisted of many nodes and developed the DRQC to improve network performance based on current and predicted queue size.

The first goal of this research was to determine the effectiveness of three types of filters to predict the router queue size in a virtual network environment using VMware Workstation. These types of filters are: the basic filter, Stuckey's extended Kalman filter, and the unscented Kalman filter. The second goal was to implement a network controller to use these traffic demand predictions, to improve the performance of networks.

5.2 Research Contributions

This research has three distinct contributions. First, this research implements Stuckey's stochastic algorithm with the extended Kalman filter in virtual network environment. Stuckey's stochastic algorithm was modeled and implemented using OPNET network simulation. This discrete-event network simulation, had limitations. This research extended the previous simulation based modeling work into a virtual network environment where the degree of traffic demand was more representative of real traffic demand with the inclusion of real operating systems and applications versus application models provided by discrete-event modeling and simulation platforms conducted in Stuckey's work.

Second, this research implemented two additional filter algorithms, the basic filter which was used as a baseline, and the UKF. Scenarios with more realistic traffic loads and two new filters allowed for better evaluation of the Stuckey's extended Kalman filter's effect on predicting the router queue size. Also, the UKF gave the best prediction performance compared with other filters.

Third, this research developed a queue controller model in virtual network environment based on the current and predicted queue size to reduce the congestion of the network.

5.3 Future Work

Suggestions for future work include:

- Adding more traffic load with different scenarios and network topologies to have a better evaluation of Kalman filter.
- Improving the Kalman filter algorithm to provide more accurate queue size prediction and therefore improve the network performance.
- Adding new features on the queue controller model such as load balancing which could be done by dynamically split flows. This will facilitate the utilization of several paths for one flow to meet its bandwidth prerequisites.
- Implementing the Kalman filter inside the router to reduce the react time or implementing it in low level languages such as C++.
- Applying queue size state prediction model in mobile ad hoc network.

References

[1] Compton, M., Hopkinson, K., and Graham, S, “The Network Tasking Order (NTO)”, *IEEE Military Communications Conference*, San Diego, CA, USA, 2008.

[2] James D. Haught, and Kenneth M. Hopkinson, Jeffrey M. Hemmes, “The Network Controller Based on Router Queue-Size Predictions”, *The JDMS, Journal of Defense Modeling and Simulation*, 2010.

[3] Nathan Stuckey, “Stochastic Estimation And Control Of Queues Within A Computer Network”, Air Force Institute of Technology, Wright-Patterson AFB, Thesis, 2007.

[4] Kim, M, “Stochastic Estimation and Control of Queues within a Computer Network”, Air Force Institute of Technology, Wright-Patterson AFB, Thesis, 2009.

[5] Pecarina, M, “Creating an Agent Based Framework to Maximize Information Utility”, Air Force Institute of Technology, Wright-Patterson AFB, Thesis, 2008.

[6] Mneimneh, S, Computer Networks: A Gentle Introduction to Queuing Theory. Hunter College of CUNY, New York. Retrieved October 2013 from: http://www.cs.hunter.cuny.edu/~saad/courses/networks/notes/note9.pdf

[7] Yan, T. and Veeraraghavan, M. Networks of Queues. Retrieved October 2013 from: http://www.ece.virginia.edu/mv/edu/715/lectures/QNet.pdf.

[8] Gill, Li, Mahanti, Luo and Williamson, C. Network Information Flow in Network of Queues. IEEE, 2008.

[9] Mohinder S. G. and Angus P. A. Kalman Filtering; Theory and Practice Using MATLAB. Wiley Publishers, 2011.

[10] Zaknich, A. Principles of Adaptive Filters and Self-Learning Systems. Springer London, 2005.

[11] Zarchan, P., and Musoff, H. Fundamentals of Kalman filtering: A Practical Approach. Books24x7.com, Norwood Mass, 2005.

[12] Randhawa, T. S., & Hardy, S. Network Management in Wired and Wireless Networks. Boston Kluwer Academic Publishers, 2001.

[13] Ward, B. The book of VMware: The Complete Guide to VMware Workstation. No Starch Press, San Francisco, 2002.

[14] In Pathan, A.-S. K. The State of the Art in Intrusion Prevention and Detection. Auerbech Publications, 2014.

[15] Davis, D. “Understanding Virtual Networking in VMware Workstation 9”. VirtualizationAdmin.com. Retrieved May 30, 2014 from: http://www.virtualizationadmin.com/articles-tutorials/vmware-server-workstation-player-articles/understanding-virtual-networking-vmware-workstation-9.html

[16] Ishtiaq Ali and Natarajan Meghanathan, “Virtual Machines and Networks Installation, Performance, Study, Advantages and Virtualization Option”, *International Journal of Network Security & Its Application (IJNSA)*, 2001.

[17] Introduction To MikroTik, Retrieved November 2013 from: http://www.broadbandbuyer.co.uk/Shop/MFR/?SupplierID=327

[18] Chui, C. K., and Chen, G. Kalman filtering: With Real-Time Applications. Springer, Berlin, 2009.

[19] Fletcher, A., and Goyal, K. Estimation from Lossy Measurements: Jump Linear Modeling and Kalman Filtering. Berkeley, CA, 2004.

[20] Goldberg, V. An Efficient Implementation of a Scaling Minimum-Cost Flow Algorithm. J. Algorithms, Academic Press, Inc. Duluth, MN, USA, 1997.

[21] Issariyakul, T., and Hossain, E. Introduction to Network Simulator NS2. Springer, New York, USA, 2010.

[22] Simon, D. Innovatia. Retrieved August 3, 2013 from: http://www.innovatia.com/software/papers/kalman.htm

[23] Smith, S., and Seiler, P. Estimation with Lossy Sensor Measurements: Jump Estimator for Jump Systems. IEEE Trans. Autom. Control, 48(12): 2163-2171, IEEE, 2003.

[24] Wolski, R. Forecasting Network Performance to Support Dynamic Scheduling using the Network Weather Service. High Performance Distributed Computing. The Sixth IEEE International Symposium on, 1997.

[25] Xia, Y., Shang, J., Chen, J., and Liu, G. Networked Data Fusion with Packet Losses and Variable Delays. System, Man, and Cybernetics, Cybernetics, *IEEE Transactions on Systems, Man, and Cybernetics*, 1107-1120, IEEE, 2009.

[26] Alberto Corigliano, Stefano Mariani. Parameter Identification in Explicit Structural Dynamics: Performance of the Extended Kalman Filter. Journal of Computational Physics, 3807-3835, 2004.

[27] Kao, J., Flicker, D., Henninger, R., Frey, S., Ghil, M., Ide, K. Data Assimilation with an Extended Kalman Filter for Impact-Produced Shock-Wave Dynamics. Journal of Computational Physics, 196:705-723, 2004.

[28] Hayder M, A. Incorporating Kalman Filter in the Optimization of Quantum Neural Network Parameters.Baghdad. Babylon University, Babylon, Iraq, 2012.

[29] Simon J. Juiler, Jeffrey K. Uhlmann. “A New Extension of the Kalman Filter to Nonlinear Systems”. *Proceeding of AeroSense (*pp. 401-422). IEEE, 2003

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 074-0188*

1. REPORT DATE *(DD-MM-YYYY)* 18-09-2014	**2. REPORT TYPE** Master's Thesis	**3. DATES COVERED** *(From – To)* Aug 2012 – Sep 2014

TITLE AND SUBTITLE Stochastic Prediction and Feedback Control of Router Queue Size in a Virtual Network Environment	**5a. CONTRACT NUMBER**
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Muflih Alqahtani, First Lieutenant, RSAF	**5d. PROJECT NUMBER**
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S) Air Force Institute of Technology Graduate School of Engineering and Management (AFIT/EN) 2950 Hobson Way, Building 640 WPAFB OH 45433-8865	**8. PERFORMING ORGANIZATION REPORT NUMBER** AFIT-ENG-T-14-S-10
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Intentionally left blank	**10. SPONSOR/MONITOR'S ACRONYM(S)**
	11.SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
DISTRUBTION STATEMENT A:
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES
This material is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

14. ABSTRACT
Modern congestion and routing management algorithms work well for networks with static topologies and moderate QoS requirements. However, these algorithms may not be suitable for modern military networks with fluid dynamic topologies and traffic demands that include mobile networks with many assets. These secure networks require a high level of Quality of Service (QoS) that must adapt to changing demands with no service interruptions. The idea of a network weatherman was developed by Stuckey to predict router queue size by using an extended Kalman filter. He modeled and exercised his design in OPNET with positive results. The goal of this research is to investigate the use of queue size predictions and network weatherman and to determine the effectiveness of three types of filters to predict future traffic demand in a virtual network environment. These filters are an extended Kalman filter, an unscented Kalman filter, and a basic filter. These queue size predictions will be used to implement a network controller to improve the performance of information technology (IT) networks and formulate some type of context awareness and cognitive process in the management of networks by reducing packet loss.

15. SUBJECT TERMS
Kalman filter, Stochastic algorithm, MATLAB, VMware Workstation, Queue, Prediction of Queue Size, Queue Controller.

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** LTC Robert J. McTasney, AFIT/ENG
a. REPORT U	**b. ABSTRACT** U	**c. THIS PAGE** U	UU	90	**19b. TELEPHONE NUMBER** *(Include area code)* (937) 255-3636, ext 4460 (robert.mctasney@afit.edu)

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39-18